Reliability Solutions in Wireless Sensor Networks

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy

Computer Science

by

Yafeng Wu

December 2011

APPROVAL SHEET

The dissertation

is submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Yafeng Wu

AUTHOR

The dissertation has been read and approved by the examining committee

[illegible] (ADVISOR)

[illegible]

[illegible]

[illegible]

[illegible]

Accepted for the School of Engineering and Applied Science:

J. H. Aylor

Dean, School of Engineering and Applied Science

January

2012

Abstract

Wireless sensor networks (WSN) composed of large numbers of small devices, are applicable to a wide range of mission-critical applications, including firefighting and emergency response, infrastructure monitoring, military surveillance, and medical applications. For these systems, reliability is the most common and important requirement. However, it is extremely challenging to achieve high-level reliability in these mission-critical applications because of resource-limited sensor devices, high-volume and burst traffic, wireless noise and interference, and unpredictable environment change and human intervention.

In this dissertation, we focus on developing new methodologies to address reliability issues in both networking and application levels. In the networking level, we propose to develop new protocols with two advanced networking techniques to improve communication reliability. The first technique is multi-channel communication. We propose a set of novel multi-channel protocols, which exploit simultaneous transmissions on different channels to reduce interference and then improve communication reliability. The second technique is Active Collision Recovery (ACR). The proposed protocol, ACR, can efficiently recover corrupted packets when collisions occur. In the application level, our work focuses on how to guarantee and maintain application-level operation correctness in mission-critical WSNs over long lifetimes. The core of our proposed solutions is a new design principle, run-time assurance (RTA). RTA requires that a system can be validated and demonstrated (periodically or on demand) that it is capable of operating correctly at run time. We develop a highly automated methodology and novel runtime framework that self-tests WSNs for RTA. In this framework, rigorous specification of both the functional logic of the application as well as the runtime assurance requirements are used to automatically generate the code for both the system implementation as well as the runtime assurance tests. The WSN then performs self-tests to validate its functions at run time. When tests fail, the WSN can automatically send alarms, and collect traces for future diagnosis and repairs.

The work in this dissertation paves the way for networking tiny and resource limited sensor nodes into high-confidence systems for mission-critical applications. Models and protocols we develop in the networking layer lead to more efficient and reliable wireless communication so that high-volume and burst data streams can be efficiently transferred across networks in a timely manner. New design principles, methodologies and a highly automated framework for runtime assurance help the designer and users verify an application's integrity at runtime, and build reliable and trustful WSN systems. In general, by developing novel protocols and methodologies in networking and application levels, an overall reliability enhancement of wireless sensor systems is achieved, which sets up a solid basis for developing wireless sensor networks technology to more mission-critical applications.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Professor John A. Stankovic, for his guidance and support for my Ph.D study and research, for his patience, movitation, ehthusiasm, and immense knowledge. I am also grateful that he nurses his students with love. He setups a model for me to be a good researcher.

I would also like to thank my committee members, Professor Sang H. Son, Professor Kamin Whitehouse, Professor Alfread Weaver and Professor Stephen D. Patek for their guidance and valuable comments on this dissertation.

It is really a privilege to work with a group of outstanding colleagues. I received a lot help from them. I would like to thank them all: Jingyuan Li, Krasimira Kapitanova, Gang Zhou, Shan Lin, Anthony Wood, Leo Selavo, Zhihen Xie, Qiuhua Cao, Jingbin Zhang, Hengchang Liu, Qiang Li and Vijay Srinivasan.

Last but not least, I would like to thank my family. Without the understanding and unconditional support from my wife Han Yu, this dissertation would not have been possible. The love from my parents, although thousands of miles away, has always been the momentum that pushes me to pursue knowledge and truth. I dedicate the dissertation to all of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Reliability in Wireless Sensor Networks

Wireless sensor networks (WSN) composed of large numbers of small devices are being investigated for a wide variety of applications. Two key advantages of these networks over more traditional sensor networks are that they can be dynamically and quickly deployed, and that they can provide fine-grained, close and continuous sensing and detecting. WSN systems are currently being used for many mission-critical applications. Automatic fire detection systems using self-organized WSNs are deployed in the context of residential and outdoor environments [9,13,14]. Sensor networks are used for chemical event tracking and pollution monitoring and control [12,89,100]. WSNs are also applied to monitor the health of civil structures like bridges and buildings [22, 55, 77, 101]. Dedicated WSNs have also been developed and used for military tracking and surveillance [38, 110]. In these applications, WSNs monitor, detect, and track abnormal events, hazards, and high-cost environmental events, and send alarms or report urgent information. Failures of these systems cause severe consequences in terms of human deaths, property loss, disaster, and military defeat.

These are just a sample of a wide range of sensor network systems that have already been built and deployed for mission-critical applications in the real world. For these system, reliability is the most important and common requirement. According to IEEE's definition, reliability is ”the ability of a system or component to perform its required functions under stated conditions for a specified period of time” [85]. Consider a firefighting system as an example of such mission-critical systems.

Low cost sensor nodes can be embedded into large city skyscrapers to support fire detection and reaction. Such systems must reliably detect a fire on any floor, activate alarms, notify fire stations, and announce and illuminate egress routes. Such systems require high reliability in their operations. However, this is a very difficult problem, e.g., there may be 100 floors, with over a 100 nodes per floor deployed densely and work as an aggregate. When detecting fires, nodes generate various sensing readings, such as temperature, humidity, smoke and acoustic, and must upload these high-volume data to the sinks reliably and timely. The system is expected to operate for many years before it must be replaced, but its functionalities may be impaired unexpectedly. Software faults and hardware failures may occur spontaneously or by human intervention. Wireless noise and interference may change dramatically as new wireless technologies are developed and deployed in or near a building, and sensor readings and network topology may change as the occupancy, activities, and equipment in a building change over time.

Like this firefighting system, many surveillance and tracking systems, industrial plants, pollution monitoring, and medical applications share the common characteristics. First, these systems are usually large and dense systems, consisting of many nodes deployed closely to cover fields of interests. Second, nodes in these systems usually possess limited CPUs, memories, and sensors all driven by small batteries. Also, they are equipped with less powerful radio components and share the limited bandwidth with neighboring nodes. Third, mission-critical systems are usually event-driven and generate high volume data and burst traffic. Forth, these systems, deployed indoor or fields, also suffer the uncertainty of impacts of unpredictable environment and human factors.

1.2 Problem Statements

WSNs features impose challenges on the design of reliable mission-critical WSNs, and make the reliability extremely difficult to achieve, especially in the following two aspects.

- *Network Level*: Protocols are needed to provide highly reliable, low-latency and robust communication in large and dense WSNs, and address collision and congestion issues caused by high volume and burst traffic.

- *Application Level*: New methodologies are needed to guarantee and maintain expected operations over system lifetimes and address issues including: unexpected Software error and faults, hardware failures or degradation, energy depletion of devices, location and time synchronization errors, sensor calibration errors that occur over time, harsh and frequently changed environment and unpredictable human intervention.

1.2.1 Solutions for Reliable communication

Communication is one of the most important and fundamental components in WSN systems. In a mission-critical WSN system, the networking component must provide reliable data communication. Otherwise crucial data and messages will be missed during transmission, cooperation among nodes will be messed up, and then systems will fail to provide its critical services. Furthermore, mission-critical applications usually also demand high-quality communication with low latency, energy efficiency and robustness. However WSN communication suffers limited bandwidth, intensive collisions and interference, the dynamics of link quality, topology and traffic. These problems become even severe in many mission-critical systems, because of their high-volume and burst traffic triggered by events, high node densities, unattended deployment and unpredictable environment. Even lots of protocols have been proposed to address these issues, but few protocols can (1) maintain high throughput and reliability with high-volume and burst traffic in dense networks, and (2) efficiently reduce and recover packet collisions. In this work, we propose to use two practical techniques to address these issues, which are multi-channel communication and active collision recovery.

1.2.1.1 Multi-channel Communication

One practical solution to maintain reliable and high-throughput traffic is to provide parallel communication with multi-frequency media access control. On the hardware side, MICAz and Telos have already provided 16 well separated frequencies [41] and multi-frequency specification has also been incorporated into IEEE 802.15.4 standard [44]. Furthermore, several multi-channel MAC protocols have been proposed to improve throughput in WSNs. However, our empirical studies demonstrate that current WSN hardware and development environment introduce practical issues to multi-channel communication, which include insufficient available channels, overhead of channel switching and coordination, and sensitivity of time synchronization errors. These reality issues make the current multi-channel MAC protocol not suitable to mission-critical applications, which feature high dense networks with burst traffics.

Therefore, these is a need to design multi-channel protocols that fit hardware reliability of WSNs and provide reliable and efficient parallel communication in dense WSNs.

1.2.1.2 Efficient Packet Collision Recover

Packet Collision occurs when two or more close stations attempt to transmit a packet at the same time. This can result in packet loss and impede network performance. Collisions become even severe in mission-critical WSNs for two reasons. First, many such systems are dense and generate burst spatially-correlated traffic, where multiple sensors in the same neighborhood all have messages to send simultaneously in response to the same event. Second, WSNs are typically equipped with few sinks to which packets from sensors converge. Such convergence cause many collisions around sinks, described as the “funneling effect” [3].The consequences of packet collisions are serious to mission-critical WSNs. Collisions can cause the loss of critical data or control information, reduce the system reliability and lead to application failures.

Many CSMA based MAC protocols are proposed in Wireless Sensor Network (WSNs) to avoid collisions, such as B-MAC [80], X-MAC [15]. These protocols can efficiently reduce collisions, but intrinsically cannot eliminate all collisions, because of hidden terminal problems, as well as collisions when multiple nodes sense the medium free at the same time. Previous work [47] demonstrates that CSMA based protocols cannot avoid such collisions when the number of concurrent transmissions grows.

Since collisions cannot be eliminated, packet recovery methods must be used. Traditional Automatic Repeat Request (ARQ) method uses acknowledgment and retransmissions for packet recovery. However, ARQ brings high latency due to waiting for ACKs, and retransmissions cause more collisions, especially with heavy traffic. New packet recovery methods exploit the Partial Corrupted

Pattern (PCP), where only some bits are actually corrupted in a "lost" packet. For instance, Forward Error Correction (FEC) helps receivers recover such partially corrupted packets by sending packets with redundant bits. As shown in [4], it significantly reduces retransmissions and delivery latency. However, since these protocols are only used to recover corrupted packets caused by lossy links, it is not suitable to recover packets corrupted by collisions in WSNs. Therefore, it is imperative to design new methodology to efficient recover broken packet once collision occurs in dense WSNs with burst traffics.

1.2.2 Solutions for Application-Level Reliability

WSNs are being widely deployed for mission-critical applications. However, they face a serious dilemma. On one side, these WSNs must operate and provide their services reliably and continuously over long time durations due to the high cost of system failure. On the other side, WSNs are very error-prone because of the use of resource-limited sensor devices, the unattended nature, and the ever changing and unpredictable environment. Furthermore, a WSN system usually consists of many components. Even each component employs fault-tolerant or other mechanism to achieve high reliability. The system may still fail because of wrong or unexpected synergistic behaviors. To address this dilemma, there must be some methodologies that directly guarantee and maintain the application-level reliability of WSN systems.

We believe that the key of address this issue is a new design principle, Run Time Assurance (RTA) that requires that systems can be periodically validated to demonstrate their run time operability and the integrity of their application semantics. The essence of run time assurance (RTA) lies in the systems capability to identify failures at the application-level on a periodical basis or by request. If a system has such an assurance capability, system administrators can utilize that to evaluate the system and make decisions whether it is necessary to take actions to repair and recover systems. We believe that RTA can be used to provide application-level reliabilities, and should be addressed as a first design principle for mission-critical systems. However, it is very challenging to design and implement WNSs with RTA, due to great variety of WSN application requirements and operation environments, as well as the complexity of RTA itself. Therefore, there is a need for general RTA design principles, guidelines and procedures, and a practical and easy-to-use framework that helps and automates implementation of various RTA WSN systems.

1.3 Contribution

The overall result of the dissertation is significant progress towards designing and developing more reliable sensor networks for mission-critical applications. Our contributions in developing new reliable methodologies can be summarized as:

- *Multi-channel Communication:* In wireless sensor network research, we are the first to investigate multi-channel realities through empirical experiments and asset its impact on existing multi-channel protocols. We identify three key practical issues for WSN multi-channel designs, which are insufficient available channels, overhead of channel switching and coordination, and sensitivity of time synchronization errors. This study provides important guidelines for multi-channel design in WSNs, and can benefit developing more sophisticated multi-frequency parallel communication for sensor networks in the future.

  We propose a novel idea to use multiple channels in WSNs, in which we divide the whole network into sub-tree and assign one channel to each tree. With this tree-level channel assignment scheme, nodes can communicate with sinks using one single channel through their own sub-trees. This new multi-channel methodology eliminates the need of time synchronization and the complexity and overhead for channel switching and coordination, and separate channel assignment from MAC-layer design and provides simplicity and flexibility to use any existing MAC protocols, while still enjoying the benefit from parallel communication of multiple channels. Furthermore, this scheme is very adaptive to the network density and the number of available channels, and can still exploit the multi-channel parallelism in high-density networks with few channels to use.

  Next, we propose network partition and channel assignment algorithms to further optimize network metrics. We first study an optimization problem which aims to find the best assignment that minimizes interference within sub-trees. We prove it a NP-hard problem and propose a greedy algorithm that yields near-optimal results. Furthermore, we take the link diversity into consideration, and propose an efficient algorithm to minimize interference as well as to meet end-to-end reliability requirements in networks with low-quality links.

We implement a novel multi-channel protocol, TMCP in both simulation and real MicaZ testbed, which combine aforementioned ideas and algorithms. Our development and evaluation demonstrates that 1) TMCP efficiently improve network performance over other multi-channel protocols with less interferences, much higher packet delivery rates, and higher network throughput; 2) TMCP works well with high network density, small numbers of available channels and poor and diverse link qualities. As a result, TMCP is a efficient and practical multi-channel solution to provide reliable communication for mission-critical solutions.

At last, we extend our multi-channel research to consider how to dynamically assign channels according to live traffic patterns at run-time. As an early work, we propose a simple and efficient traffic-aware channel assignment scheme based on the assumption that where perfect information about current and future traffic is available. The new scheme exploits this information to minimize interference occurring with real traffic. We compare this scheme with two typical static channel assignment schemes by simulation, and results show that being traffic-aware can substantially improve the performance of channel assignment. This baseline analysis helps establish the potential benefits of traffic-aware channel assignment algorithms.

- *Efficient Collision Recovery:* We are the first to conduct an empirical study on bit error patterns of collided packets in WSNs. Our study reveals that when nodes send packets with two different sizes, only certain bits in the long packet are corrupted in collisions between long and short packets, and this chunk of erroneous bits has almost the same size of the short packet. We call such collisions LS-collisions. LS-collisions produce a regular PCP in collided packets, which can be exploited to achieve efficient collision recovery.

  In order to justify this idea, we provide a thorough analysis based on the model derived from scenarios where N senders compete for transmission using CSMA. The analysis demonstrates that by exploiting LS-collisions, we can achieve a higher probability of successful transmission, as well as a much higher transmission efficiency defined as the ratio of successfully transmitted information bits to overall bits transmitted. The analysis also gives insights on how to choose the right proportion of nodes sending long packets and short packets, and a sub-optimal probability distribution for senders to randomly select contention slots that maximizes the above two metrics.

We then present ACR, an Active Collision Recovery protocol that mitigates the negative impact of collisions by efficiently recovering collided packets. Unlike other existing recovery schemes, ACR is ”active” that it actively transforms potential collisions into LS-collisions to maximize the recovery probability, rather than passively waiting for collided packets to be recovered. ACR then uses a block based FEC scheme to efficiently recover corrupted packets due to LS-collisions. To identify erroneous blocks, ACR employs a novel RSSI-based error detection method, which does not introduce extra checksum overhead. In case of recovery failures, ACR also has a backup ARQ scheme. Unlike ZigZag [36] or PPR [46], ACR does not require customized hardware. ACR can be easily integrated into existing CSMA protocols with off-theshelf sensor devices.

We implement and evaluate the ACR prototype on a Tmote testbed. Results demonstrate that ACR achieves 25% higher transmission efficiency than existing recovery schemes.

- *Runtime Application-Level Assurance:* Continuous and reliable operation of WSNs is notoriously difficult to guarantee due to hardware degradation and environmental changes. We propose and demonstrate a methodology for *run-time assurance* (RTA), in which we validate at run time that a WSN will function correctly, irrespective of any changes to operating conditions since it was originally designed and deployed. The basic approach is to use program analysis and compiler techniques to facilitate automated testing of a WSN at run time. The developer specifies the application using a high-level specification, which is compiled into both (i) the code that will execute the application on the WSN, and (ii) a set of input/output tests that can be used to verify correct operation of the application. The test inputs are then supplied to the WSN at run time, either periodically or by request. The WSN performs all computations, message passing, and other distributed operations required to produce output values and actions, which are compared to the expected outputs. This testing process produces an end-to-end application level validation of all components required for correct system operation, including hardware, software, network topology, and any interactions between them.

We have implemented a framework for designing and automatically testing WSN applications using the RTA methodology. The developer specifies the application using a high-level Sensor Network Event Description Language (SNEDL) [50], which is an extended Petri net model. Our system compiles the SNEDL model down to TinyOS [61] code that runs on the Telos nodes [24], as well as tests the defined mappings between sensor input values and system outputs.

We evaluate our implementation by designing a fire detection system and executing it on a network of 21 Telos nodes. We artificially introduce failures into the system, including node failures and location errors, and compare the performance of RTA to that of an existing health monitoring solution [86]. Our results indicate that RTA misses 75% fewer system failures and also produces 70% fewer maintenance dispatches than health monitoring.

Our work in this dissertation paves the way for networking tiny and resource limited sensor nodes into high-confidence systems for mission-critical applications. Models and protocols we develop in the networking layer lead to more efficient and reliable wireless communication so that high-volume and burst data streams can be efficiently transferred across networks timely. New design principle, methodologies and highly automated framework we propose in runtime assurance help designer and users verify an application's integrity at runtime, and build reliable and trustful WSN systems. In general, by developing novel protocols and methodologies in networking and application levels, an overall reliability enhancement of wireless sensor systems is achieved, which sets up a solid basis for developing wireless sensor networks technology to more mission-critical applications. Four major conference publications [107] [107] have come from this dissertation.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows: Chapter 2 is a detailed survey of the state-of-the-art of research work on sensor networking and application-level reliability we explore in this dissertation. Chapter 3 presents multi-channel reality found in WSNs, a set of multi-channel algorithms and protocols that exploit multi-channel to improve network reliability and efficiency. Chapter 4 presents a novel efficient packet collision recovery scheme and how this scheme improve communication reliability. Chapter 5 present Run Time Assurance concept and principles as well as an automated framework that help build WSNs application using the RTA methodology. Finally, Chapter 6 concludes the dissertation and proposes potential research extensions.

Chapter 2

State of the Art

Publications on related work are introduced in this chapter. Section 2.1 describes the state of the art for multi-channel communication. Section 2.2 describes the state of the art for packet collision detection and recovery. Section 2.3 explains the related work on application-level assurance.

2.1 State of the Art for Multi-channel Communication

In the general wireless network, frequency diversity has been studied for years and a significant number of multi-frequency MAC protocols have been proposed. However, these protocols are not suitable for typical WSN applications. First, to save energy and reduce product cost, each sensor device is usually equipped with a single radio transceiver. This single transceiver cannot transmit and receive at the same time, nor can it function on different frequencies simultaneously. This restricted hardware is quite different from more powerful hardware assumed in other wireless systems. For example, protocols [94] [98] are designed for frequency hopping spread spectrum (FHSS) wireless cards, and protocol [27] assumes the busy-tone functionality on the hardware. In protocols [74] [104] [73] [19], the hardware is assumed to have the ability to listen to multiple frequencies at the same time. Second, the network bandwidth in WSNs is very limited and the MAC layer packet size is very small, 30$\sim$50 bytes, compared to 512+ bytes used in general wireless ad hoc networks. Due to the small data packet size, the RTS/CTS control packets in IEEE 802.11 [43] no longer constitute a small overhead that can be ignored. So protocols [8] [83] [1] [32] [62] that are based on IEEE 802.11, and protocols [91] [45] [98] [94] that use RTS/CTS for frequency negotiation are not suitable for WSN applications, even though they exhibit good performance in general wireless ad hoc networks. Different from all the above solutions.

Instead of assigning different frequencies to neighboring nodes in an infrastructureless wireless sensor network, in an infrastructure based cellular network, different frequencies are assigned to different cellular towers that use very high sending powers and usually form a hexagon graph. In a cellular network, the frequency assignment is normally formulated as a multicoloring problem on a varying weight graph [48], and distributed algorithms are proposed to address it. To improve flexibility and reduce computational complexity, [11] models it as an optimization problem. It first provides a localized solution, then extends the solution to a distributed one by ensuring mutual exclusion. In [23], a generic solution template is proposed and applied to cellular frequency assignment for temporary tasks with a load balancing constraint. Since all these are developed for high power, infrastructure based, and normally hexagon topology cellular networks, they do not directly address the hardware and application challenges in wireless sensor networks.

In wireless sensor networks, media access control has received intense research attention, and a number of solutions have been proposed [2] [16] [113] [53] [80] [112] [81] [26] [102] [30]. While these solutions work well when one physical frequency is used, parallel data transmission when multiple frequencies are available is not considered.

Recently, MMSN [118], TMMAC [115] and MCMAC [21] are three new multi-channel MAC protocols designed especially for WSNs. They all try to assign different channels to nodes in a two-hop neighborhood to avoid potential interferences. We call these node-based multi-channel protocols. Simulation results show that they improve performance in WSNs compared with single channel protocols. However, with nodebased channel assignment schemes, a node typically has a different channel from its downstream and upstream nodes. Within a multi-hop flow, nodes have to switch channels to receive and forward packets which can cause very frequent channel switching and potential packet losses. In order to avoid such packet losses, node-based protocols use some negotiation or scheduling schemes to coordinate channel switching and transmissions among nodes with different channels. For instance, all three protocols mentioned above use time slots to coordinate transmissions. They face practical issues in real WSNs, including: 1) a large number of orthogonal channels are needed for channel assignment in dense networks; 2) they require precise time synchronization at nodes, 3) channel switching delay and scheduling overhead cannot be ignored because of frequent channel switching, especially for high data rate traffic, and 4) these protocols are typically complex, which require more resources at motes. Our paper studies these practical issues through empirical experiments with Micaz motes and shows that node-based protocols may not be suitable for WSNs in practice.

We are also aware that an industrial solution called TSMP [28] exists, which, instead of using frequency assignment, requires a connection survey of entire network on every physical frequencies. Rather than developing MAC layer solutions, [108] and [103] propose to use frequency diversity to defend jamming attacks, and [59] use frequency diversity as well as control theory to optimize network throughput in a forest based data collection application.

More recently, two different channel assignment methods are proposed. A component-based protocol is presented in [99] which assigns channels to connected components in wireless ad hoc network, and in [60], nodes dynamically select channels based on a control theory approach to achieve load balance among channels. While these solutions have a similar favor in channel assignment to us, our scheme focuses on how to use multi-channels to construct the optimal topology with low interferences and optimize throughputs in practical WSNs.

2.2 State of the Art for Packet Collision Recovery

Packet Collision occurs when two or more close stations attempt to transmit a packet at the same time, which results in packet loss and impede network performance. Many CSMA based MAC protocols are proposed in Wireless Sensor Network (WSNs) to avoid collisions, such as B-MAC [80], X-MAC [15]. These protocols can efficiently reduce collisions, but intrinsically cannot eliminate all collisions, because of hidden terminal problems, as well as collisions when multiple nodes sense the medium free at the same time. Previous work [47] demonstrates that CSMA based protocols cannot avoid such collisions when the number of concurrent transmissions grows.

Another way to avoid collisions is to apply multiple frequencies to eliminate transmission interferences, which is the topic of Chapter 3. However, our previous results demonstrate that multi-channel protocols cannot remove all collisions in dense WSNs, due to insufficient channels and channel coordination costs. When collisions still occur, a packet recovery method should be applied to recover lost packets and ensure end-to-end delivery. Traditional Automatic Repeat Request (ARQ) method uses acknowledgment and retransmissions for packet recovery. However, ARQ brings high latency due to waiting for ACKs, and retransmissions cause more collisions, especially with heavy traffic. New packet recovery methods exploit the Partial Corrupted Pattern (PCP), where only some bits are actually corrupted in a "lost" packet. For instance, Forward Error Correction (FEC) helps receivers recover such partially corrupted packets by sending packets with redundant bits. As shown in [4], it significantly reduces retransmissions and delivery latency. However, since these protocols are only used to recover corrupted packets caused by lossy links, it is still unclear if such methods can be used to recover packets corrupted by collisions in WSNs.

Two novel partial packet recovery schemes have been recently proposed to recover corrupted packets from collisions, which are the PPR protocol [46] and Zigzag decoding [36]. However, both techniques need support from customized hardware. For example, PPR needs the hints from radio chips to detect which codeword is corrupted, and Zigzag implementation modifies modulation modules to recover bits from errors. They do not directly apply for the widely used off-the-shelf sensor devices like MicaZ and TelosB.

2.3 State of the Art for Runtime Application-Level Assurance

Our work is highly related to, but also distinct from a number of research areas including testing, fault tolerance, self-healing, debugging, health monitoring, and system management.

Although testing has always been a major part of the development of software applications, very limited amount of work has been done in the area of testing WSN applications. A few characteristics of WSN applications such as operating in concurrent, event-based, and interrupt-driven manner considerably complicate the development of code representation. Nguyen et al. [75] proposed application posting graphs to represent behaviors of WSN applications. Regehr [84] designed a restricted interrupt discipline to enable random testing of nesC programs. Lai et al. [57] study inter-context control-flow and data-flow adequacy criteria in nesC programs. However, all previous work is intended for testing applications prior deployment when the size of the test suite is not as critical. Therefore, WSN-specific approaches for decreasing the size of the number of necessary tests have not been considered until now.

There is a great array of fault tolerance and reliability techniques developed over the last 50 years many of which have been applied to WSNs [71, 78, 87, 93]. We expect that any WSN that must operate with high confidence will utilize many of these schemes. However, most existing approaches, such as eScan [109] and CODA [18], aim to improve the robustness of individual system's component. Therefore, it is difficult to use such methodologies to validate the high-level functionality of the application. Similarly, self-healing applications [37,38,42], although attempting to avoid violations of RTA, are not capable of demonstrating adherence of the system to key high-level functional requirements.

Debugging is a complicated process for WSNs and many different approaches exist. Marionette [31] and Clairvoyant [111] are source-level debuggers allowing access to source-level symbols. MDB [92] supports the debugging of microprograms. SNTS [70], Dustminer [52], and LiveNet [7] use overhearing to gain visibility into the network operations. Some debugging approaches are based on invariants [40], and others attempt to use data mining to discover hard to find bugs [68]. EnviroLog [66] uses a *record and replay approach* where it stores and replays the I/O on the sensor nodes. However, all of these debugging mechanisms are either used prior deployment or in a *post mortem* manner, where data about the application is collected and then analyzed offline. These techniques do not provide a way to monitor and analyze the application's behavior at run time.

Many applications have been developed to achieve sensor network hardware verification. Sympathy [82], for example, is concerned with routing problems, Health-monitoring systems such as MANNA [58], LiveNet [7], and Memento [86] employ sniffers or specific embedded code to monitor the status of a system. However, such applications monitor low-level components of the system instead of high-level application requirements. Therefore, they cannot be used as a substitute for our RTA framework. Instead, if available, such applications could be used as a monitoring component. Information from these systems could be used for RTA checks or to activate further system state checks.

There are very few overall system-management systems for WSNs [5, 97]. Most of them manage only a few system properties such as energy [49], topology, or bandwidth. However, we have not found any that address RTA for high confidence systems in terms of application-level requirements.

Chapter 3

Multi-Frequency Communication

As an emerging technology, Wireless Sensor Networks (WSNs) have a wide range of potential applications, including many mission-critical applications, such as intruder detection [10], coal mine surveillance [63], and forest fire detection [114]. In these applications, a large burst of packets is generated due to a change in the monitored conditions and needs to be transported in a reliable and low-latency manner to a base station, which is very difficult to achieve due to the limited bandwidth, intensive collision and congestion. A large number of single-channel communication protocols have been proposed for the MAC, routing and transport layers with the objective of energy efficiency. They are suitable to traditional low-duty cycle and low-date rate applications, but cannot provide reliable communication to support high-rate traffic in large and dense networks. For example, in the "Ears on the ground" project [116], the network fail to transmit multiple acoustic streams to the sink with single-channel protocols.

Multi-channel communication is an efficient method to eliminate interference and contention on the wireless medium by enabling parallel transmissions over different frequency channels [56]. Newer generations of commercially available radios used on the sensor nodes support multi-channel communication. For example, the Nordic NrF905 radio [76] used on Ambient μNodes [6] and the CC2420 radio [20] for MICAz [25] and TelosB [96] sensor nodes can be tuned to operate on different frequencies. Hence, there exists the potential to use multi-channel communication for alleviating the effects of competitive communication environment in WSNs. In this chapter, we focus on how to achieve reliable and low-latency delivery of high-rate traffic in bandwidth limited WSNs by making use of the multi-channel capability of the sensor radios. We have conducted extensive research on this topic [106] [120], and we answer the following research questions in this chapter:

- What are the hardware realities and characteristics of multi-channel communication in WSNs? What are the advantages and practical challenges for multi-channel protocol designs?
- How to assign the small number of available channels to nodes in dense WSNs such than we can eliminate potential interferences as much as possible?
- How to eliminate the packet loss due to channel mismatch between senders and receivers? How to efficiently avoid or remove overhead of channels switching and coordination?
- How to assign channels in dense networks with highly diverse link qualities? How to select reliable routes in multi-channel communication?
- How to effectively assign channels among nodes with different traffic load? How to utilize nodes' traffic information to further improve communication performance?

This chapter is organized as follows: In Section 3.1, we present empirical results from experiments that investigate multi-channel realities found in WSNs. Next, we propose and evaluate a novel tree-based multi-channel protocol (TMCP) in Section 3.2. In Section 3.3, we present and evaluate an extension of TMCP which maintains high reliability in networks with diverse link qualities. Furthermore, we present a traffic-aware channel assignment scheme in Section 3.4. Finally, in Section 3.5, we give the conclusions and describe the future work.

3.1 Experiments on Multi-channel Reality

In order to design good protocols, we need to better understand multi-channel realities in WSNs. In this section, we first conduct a set of empirical experiments to investigate multi-channel interference properties of Micaz hardware, including channel switching time, adjacent channel interferences and interferences with 802.11 networks. These properties are well studied in wireless ad hoc networks [119] [79] [72], but there is a lack of empirical studies in WSNs. Then we measure the performance of node-based multi-channel schemes on a single path and investigate the impact of time synchronization errors. With these experimental results, our analysis shows that current node-based schemes are not suitable for dense and large sensor networks, as well as applications with high data rates.

3.1.1 Channel Switching Time

Channel switching time is measured as the time length it takes for motes to successfully switch from one channel to another. More importantly, we want to measure how long motes wait to send a packet with a new channel after finishing the last transmission in its previous channel. This parameter impacts the maximum network throughput, because nodes cannot receive or send any packets during this period of time, and it also affects the efficiency of channel sensing, where motes need to sense through available channels. Works in [56] shows that the network bandwidth degrades as a function of $s/(s+t)$ where s is the switching time and t is the transmission time. However, it is lack of experiments to measure this value on real motes. For example, TMMAC in [115] assumes that the switching time is $80\mu s$, and uses it in simulations, but in [90] it is claimed that the switching time is $300\mu s$ with the CC2420 chip. In this subsection we measure the channel switching time of Micaz motes [25]. In our experiments, one mote alternately changes between Channel 11 to Channel 12. Every time after switching to a channel, it sends out a packet immediately, and changes to a new channel as soon as the transmission is finished. We also put two other motes in the transmission range of the sender, which are tuned to Channel 11 and Channel 12, respectively, to receive packets. These two motes are used to verify the success of channel switching and packet transmissions. During the experiments, the carrier sensing and back-off operations in the MAC layer are disabled. We measure the average number of packets the test mote can send in one second, denoted as $N_(1)$. In contrast, we also measure the same value of the test mote without changing channels, denotes as $N_(2)$. According to the bandwidth degradation function, we calculate the channel-switching time s as:

$$s = \frac{10}{N_1} - \frac{10}{N_2} \tag{3.1}$$

By repeating experiments 100 times, we get the average channel-switching time of MicaZ motes are: $24.3\mu s$, which is surprisingly less than previous values. We then conduct the same experiments with different Micaz motes, as well as experiments with the transmitter switching from Channel 11 to other channels. In both scenarios, the channel-switching time does not have obvious changes. (In our experiments, all values are in the range of $23.6\mu s$ to $24.9\mu s$.)

We also measure the minimum time a MicaZ mote takes to send one packet, which is around $1.3ms$ to $1.4ms$. So, the channel-switching time is only 1.8% of the transmission time and degrades the bandwidth by 1.77%. Therefore the channel-switching time does not have a significant impact on network performance even with frequent switching.

3.1.2 Number of Available Orthogonal Channels

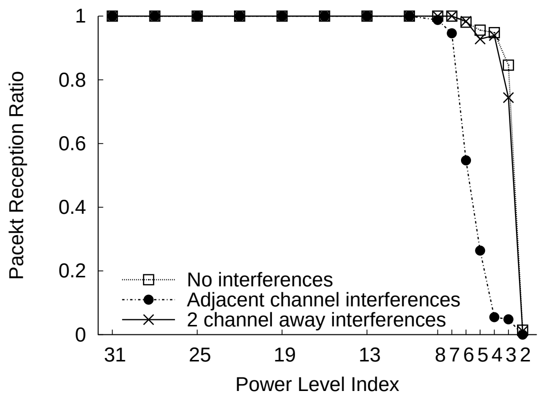


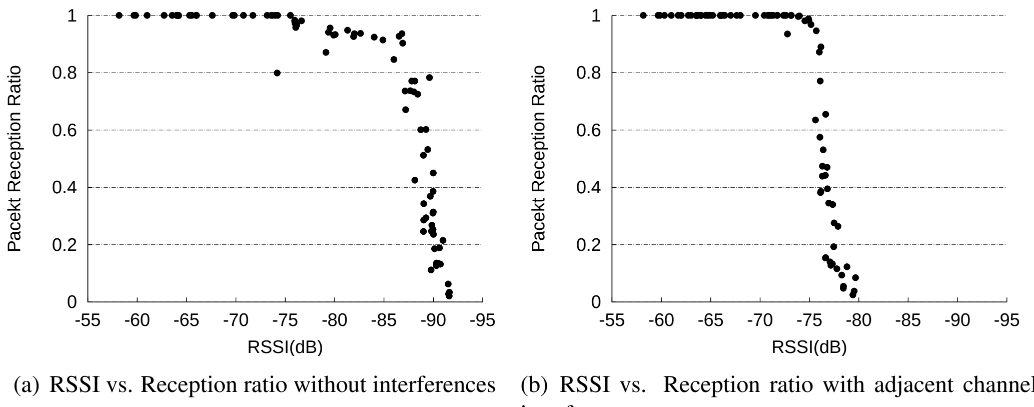
Figure 3.1: Transmission power level index vs. Packet reception ratio

An important parameter for multi-channel designs is the number of channels which can actually be used in WSNs. The CC2420 radio chip [20] used in Micaz provides 16 non-overlapping channels, with 5MHz spacing. However, not all channels can be used in a single sensor network to provide parallel transmissions because of close channel interferences and interferences caused by 802.11 networks.

3.1.2.1 Non-orthogonal Channel Interferences

Non-orthogonal channel interferences are well studied in general wireless networks [72]. For WSNs hardware, the CC2420 chip specification [20] indicates that the adjacent channel rejection is 45/30 dB, but few works study its real impact on the performance of multi-channel WSNs. In the following we present experiments to study this phenomenon.

In the first experiment, we place three Micaz motes in a line, with one transmitter, one receiver and one jammer. The jammer's transmission is synchronized with the transmitter to generate interferences. Both the transmitter and the receiver use channel 11. While the transmitter changes its transmission power, we measure packet reception ratios of the receiver in three cases, without the jammer interfering, with the jammer interfering at channel 12 (the adjacent channel interference) and at channel 13 (2 channel away). The results of this experiment are illustrated in Figure 3.1. We can see that without interferences, the receiver can maintain an above 90% packet reception ratio until the transmitter uses power levels lower than 3. However, with adjacent channel interferences, the packet reception ratio decreases to 50% when the transmission power level is below 7, which clearly shows that adjacent channel interferences greatly impact radio reception and they are not negligible. On the other hand, the curve of the two channel away interferences is very close to the one without interferences, which implies that the impact of two channel away interferences is small. We run the same experiments with other channels, and they show the similar result.



(a) RSSI vs. Reception ratio without interferences (b) RSSI vs. Reception ratio with adjacent channel interferences

Figure 3.2: RSSI and Packet reception ratios under two types of interference

In order to further investigate the impact of adjacent channel interferences, another set of experiments is conducted to determine the relation of Received Signal Strength Indication (RSSI) threshold and different channel interferences. In these experiments, we fix the positions of the receiver and the jammer, which are 2 feet apart, and the transmitter moves along the line in different places. Experiments are run in two cases, with and without adjacent channel interferences. Results are shown in Figure 3.2(a) and (b), where each data point presents a pair of RSSI and packet reception ratios. We can see that the RSSI threshold for above 90% packet reception ratio is around -87dB without interferences, while that threshold increases to -77dB with adjacent channel interferences. Transmission links with RSSI between -77dB and -87dB become unreliable when adjacent channel interference occurs.

The existence of adjacent channel interference makes it impossible to use all 16 channels in a single sensor network. Because current multi-channel protocols assume that if every node in two-hop neighborhood is assigned different channel, the whole network is collision-free. But, if two-hop neighbors use adjacent channels, collisions still occur. In order to avoid these unexpected collisions, the safe way is to only use non-adjacent channels in multi-channel protocols.

3.1.2.2 Interferences with 802.11 networks

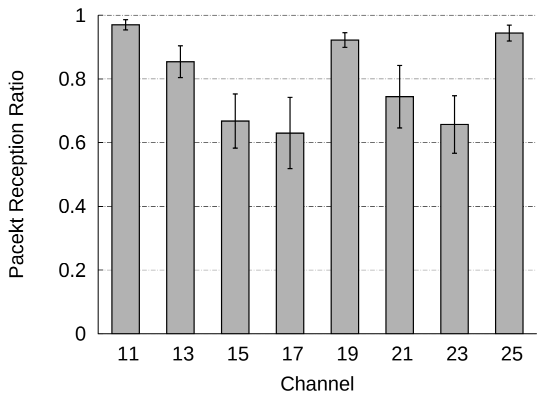


Figure 3.3: Packet reception ratios of different channels

Another factor that affects the number of available channels is the interference with 802.11 networks. 802.15.4 specification shows that one 802.11 channel can potentially collide with four

802.15.4 channels. This problem is also studied in [119] [79]. Here, we also present a simple experiment to show how 802.11 networks impact channels in WSNs. We put 8 pairs of Micaz nodes closely together in a department office, where multiple 802.11 networks exist. Each pair uses one unique channel to transmit packets within the pair. All 8 channels are orthogonal with each other. We run the experiment several times and measure the average packet reception ratios. Results are shown in Figure 3.3, with the standard deviation of each data. We can see only 3 channels (11,19,25) have good link qualities (reception ratios above 90%), and link qualities of the other 4 channels are poor (reception ratios around 60%) and unstable (large standard deviations). This experiment shows that multi-channel protocols must have capabilities to work well with a small number of available channels. Otherwise their performance may greatly degrade in such indoor scenarios.

From above experiments, we can see that even the CC2420 chip provides 16 non-overlapping channels, only a few available channels can be used simultaneously to do parallel transmission in many application scenarios. On the other side, current node-based protocols try to assign different channels to nodes in two-hop neighbors to avoid potential interference. Ideally, if we have enough channels, all in-terference can be eliminated. However, the number of available channels is far away below the number of nodes in two-hop neighbors, except in very sparse networks. For an instance, in a normal sparse network in which every node has 3 neighbors in average, the average number of nodes in two-hop neighborhood in 12, which exceeds the number of orthogonal channels. Therefore, after channel assignments, nodes within two hops may still use the same channel and cause radio collisions. In order to deal with such a problem, node-based protocols have to use more complex coordination schemes. For example, MMSN [118] adds techniques of toggle snooping and toggle transmission in each time slot. These schemes require more precise time synchronization or add more scheduling overhead, which make node-based protocol even harder to deploy in real systems.

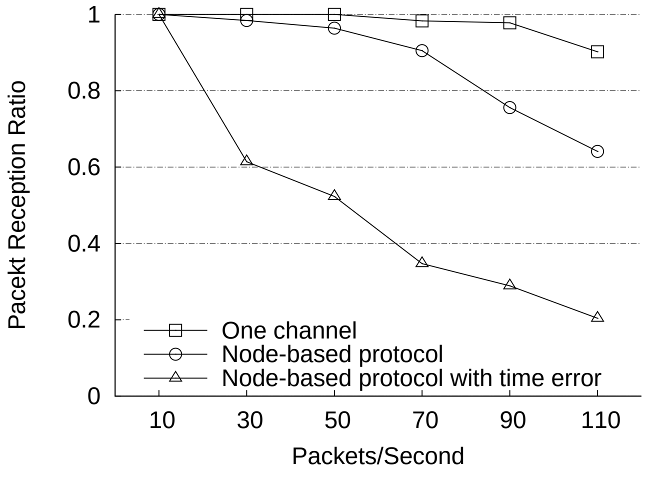


Figure 3.4: Packet reception ratio vs. Source Data rate

3.1.3 Impact of Time Synchronization Errors

Another crucial factor which can greatly impact the performance of current node-based protocols is *time synchronization error*. As mentioned before, current node-based schemes need precise time synchronization at each node to coordinate transmissions and channel switching. But, low-power Micaz motes cannot provide very high time accuracy. The clock drift of a Micaz is known to be $40ppm$ (part-per-million), which means that the clock drift can be $40\mu s$ after 1 second. In order to investigate the impact of time errors, we conduct a set of experiments on Micaz motes. We put 5 Micaz motes on a line. The first node transmits packets to the final node one-by-one hop. Each node is assigned a unique channel. At the beginning, all nodes are synchronized.

First, we use a simple time-slot based scheme as a prototype of node-based protocols. In this scheme, a time period of $10ms$ is divided into two time slots. In the first time slot, nodes in odd positions switch their channel and send packets to their next nodes, while nodes in even positions stay at their own channels and receive packets, and vice versa in the second slot. With different data rates at the source, we measure the end-to-end performance in terms of packet reception ratios. After these experiments, we wait for 10 minutes, do the same experiment again without re-synchronization and measure the second set of results, which present the performance of the node-based protocol with time errors. Finally, we modify all nodes to use a single channel, and employ the standard

CSMA protocol to transmit packets. These results are illustrated in Figures 3.4. It can be seen that without time errors, the node-based scheme always has higher packet reception ratios than the single channel scheme. The saturated packet rate (packet reception ratio is above 90%) of the two schemes are around 90msg/sec and 50msg/sec, respectively. On the other hand, with time errors, the node-based protocol has very low packet reception ratios. The saturated packet rate is around 10msg/sec, which means that the protocol can only support a low data rate for end-to-end traffic (around 3kb/s) without synchronization. This experiment confirms that node-based protocols can improve communication performance, but have large performance degradation with time errors. Furthermore, this degradation can be amplified in large and dense networks, with longer paths and more complex coordination schemes. It also shows that node-based protocols cannot provide reliable and stable communication services for high data rate traffic. One possible solution is to perform the time synchronization operation periodically. For the above experiments, nodes need to be synchronized more frequently than every 10 minutes to guarantee the performance. However, time synchronization protocols in WSNs can be costly, consuming extra bandwidth and power, which makes frequently re-synchronizing impractical, especially for high data rate mission-critical applications or for dense and larger networks.

3.1.4 Discussion on Multi-channel Realities

In this section, we studied three aspects of multi-channel reality in wireless sensor networks through intensive experiments. Our analysis focuses on how these properties impact the channel assignment strategy and current multi-channel MAC protocols. First, we find that the channel switching time is around $24\mu s$. Such short channel-switching time does not significantly decrease the bandwidth, even when nodes change channels for every packet. Secondly, we find that the number of available orthogonal channels is small because of adjacent channel interference and interference with 802.11 networks. In dense networks, it is impossible to assign different channels to neighboring nodes. With such a small number of channels, existing MAC protocols have to use complicated coordination schemes, which need highly precise time synchronization among nodes. However, our third study shows that current multi-channel protocols have large performance degradation even with small time synchronization errors. The degradation becomes even severe in dense networks with high-volume traffics, which indicates that new multi-channel protocols are needed for mission-critical applications. Our study also gives two criteria for multi-channel protocol design in sensor networks. First protocols should work well with a small number of channels. Secondly, it should not have too sophisticated coordination scheme, especially not require time synchronization or at least not require frequent synchronizations among nodes.

3.2 TMCP: A Novel Tree-based Multi-channel Protocol

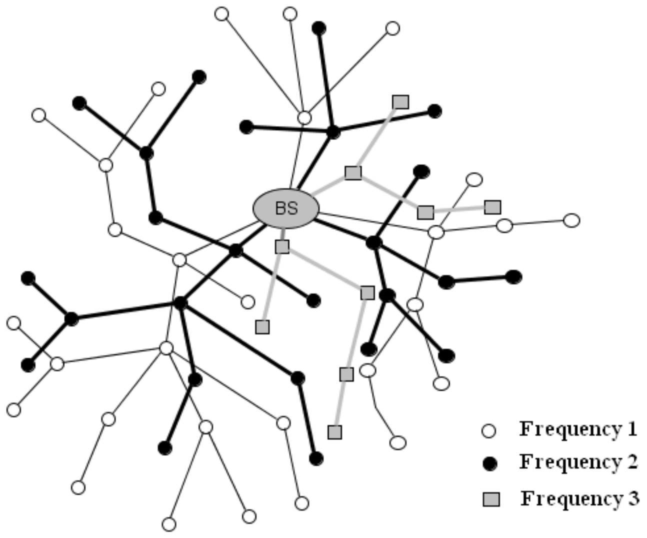


Figure 3.5: The conceptual design of TMCP

In this section, we propose a novel tree-based multi-channel protocol (TMCP) for mission-critical applications, which allocates channels to disjoint trees and exploits parallel transmissions among trees. In order to minimize interference within trees, we define a new channel assignment problem which is proven NP-complete. Then we propose a greedy channel allocation algorithm which outperforms other schemes in dense networks with a small number of channels. We also presented the evaluation results of TMCP in this section.

3.2.1 Overall Design

Every multi-channel protocol for WSNs has two main components, *channel assignment* and *transmission coordination*. As shown in section 3.1, the multi-channel realities of WSNs affect current node-based multi-channel protocols in both components. The small number of available orthogonal channels cannot satisfy the requirement of node-based channel assignment, especially for dense networks. Unavoidable time errors impact transmission coordination among nodes with different channels, especially for applications with high data rates. In order to overcome these two problems in practical networks, we believe that new multi-channel schemes should first use a coarse-grained channel assignment strategy, instead of node-level assignment, and secondly, it should try to avoid complex coordination schemes by reducing channel switching and communication among nodes with different channels.

On the other side, we also notice that sensor networks have a dominant traffic pattern, the *data collection* traffic, where multiple information flows generated at sensor nodes converge to the base-station. Currently, most data collection schemes build some tree structure connecting the base station and nodes, and then forward packets along the tree. However, with a single channel, transmission collisions within the tree and flow congestions at nodes greatly decrease the network performance.

Based on above observations, we propose a Tree-based Multi-Channel Protocol (TMCP) for data collection applications in WSNs. The idea of using multi-channel is to firstly partition the whole network into multiple vertex-disjoint subtrees all rooted at the base station and allocate different channels to each subtree, and then forward each flow only along its corresponding subtree, shown in Figure 3.5. The superiority of TMCP is two-fold. First, for practical concerns, with a coarse-grained channel assignment, it requires much fewer channels than node-based protocols. Also since every flow is forwarded in one subtree with one channel, we do not need a sophisticated channel coordination scheme, which implies that TMCP can work without the need for time synchronization. Secondly, for performance concerns, because it assigns different channels among subtrees, it can increase network throughput and reduce packet losses by eliminating inter-tree interferences and exploiting spatial reuses of parallel transmissions among subtrees.

TMCP has three components, Channel Detection (CD), Channel Assignment (CA), and Data Communication (DC). The CD module finds available orthogonal channels which can be used in the current environment. To do this, two motes are used to sample the link quality of each channel by transmitting packets to each other, and then among all channels with good link qualities, non-adjacent channels are selected. At this point we have k channels.

Given k orthogonal channels, the CA module partitions the whole network into k subtrees and assigns one unique channel to each subtree. This is the key part of TMCP. The goal of partitioning is to decrease potential interference as much as possible. We can see that after partitioning, interferences in the original network can be divided into two categories, one is the interference among different trees, called inter-tree interference, which is eliminated by assigning different orthogonal channels to each subtree, and the other is the potential interference among nodes within a tree, called the intra-tree interference. Because we assign the same channel to all nodes of one subtree, the intra-tree interference cannot be avoided in our scheme and becomes the main performance bottleneck. So, the goal of partitioning is to divide networks into subtrees, each of which has lower intra-tree interferences. In next section, we will further study this problem.

After assigning channels, the DC component manages the data collection through each subtree. When a node wants to send information to the base station, it just uploads packets along the subtree it belongs to. Here, we assume that the base station is equipped with multiple radio transceivers, each of which works on one different channel. We can see that because of the tree-based channel assignment strategy, DC is very simple without the need of time synchronization. Also, the base station can use this network structure to perform data dissemination. When the base station wants to send commands or update the code, it can send out packets through all transceivers, and then packets will go through every subtree and reach all nodes in networks.

3.2.2 Minimum Interference Channel Assignment Problem

TMCP uses a new tree-based channel assignment scheme. As mentioned earlier, the goal of this assignment scheme is to minimize intra-tree interferences. In this subsection, we formally define this problem, study its complexity, and present a greedy algorithm, and evaluate its performance by simulation experiments.

3.2.2.1 Model and Problem Definition

We assume that a sensor network is a static graph $G = (V, E)$, where V is the set of all nodes in the network, and E is the set of edges between two nodes who can talk to each other in one hop. Here, we only consider the data collection traffic in networks. Next, we define the interference value of a node in a tree. Reference [17] introduces an explicit definition of the interference value, based on the number of other nodes potentially disturbed by transmission of this node. In other words, interference is considered to be an issue at the sender instead of at the receiver. Because of the fact that the interference is actually a problem occurring at the receiver, we use a receiver-centric interference definition. The interference value of a node A is the number of other nodes by which the reception at A can be disturbed.

Definition 3.1. *The interference set of a node u should be defined as $INT(u) = \{v | u \in D(v, I_v)\}$, where $D(v, I_v)$ is the interference disk with node v in its center and radius I_v, and the interference value of a node u is defined as $int(u) = |INT(u)|$.*

Here, we assume that when a node is transmitting, all nodes within the transmitter's interference disk will be disturbed. We note that this assumption may not always be true in real networks because the interference region is not spherical observed in [117], and interference sets of nodes may change during the time. But we can use a larger interference disk to cover the actual interference region, and compute a conservative interference set for each node. We use the interference range I_v instead of the communication range R_v to describe the interference region. By the observations of [117], they are different in real networks. Furthermore, we use the assumption from the protocol interference model [56], where $I_v = (1 + \alpha) \times R_v$, and $\alpha > 0$ implies that all of u's neighbors belongs to $INT(u)$.

Next, we define the intra-tree interference value of a tree. There are two concerns. First, we should use the maximum interference value I_{max} as the interference value of the tree. Given the bandwidth B at each node, it can be proved that the theoretical lower bound of the single-flow capacity in this tree is B/I_{max}. Thus, I_{max} decides the lowest data rate of a single flow through the tree, which is important for applications. Secondly, since our interference model is receiver-centric and leaf nodes are not receivers in data collection traffic, the interference of a tree is the maximum interference value among all *non-leaf* nodes.

Definition 3.2. *The intra-tree interference value of a tree T is defined as $int(T) = \max\{int(u) : u$ is a non-leaf of $T\}$*

As an example, the intra-tree interference value of the tree in Figure 3.6 is 4, in spite of the fact that there is a leaf node with the interference value of 5. Here, we want to emphasize that dealing with the non-leaf condition is not trivia. In fact, it implies that if a node has a large interference value, we can set it as a leaf and then it is not needed to receive packets from other nodes in the data collection traffic. By doing this, we can indeed reduce the interference in the tree.

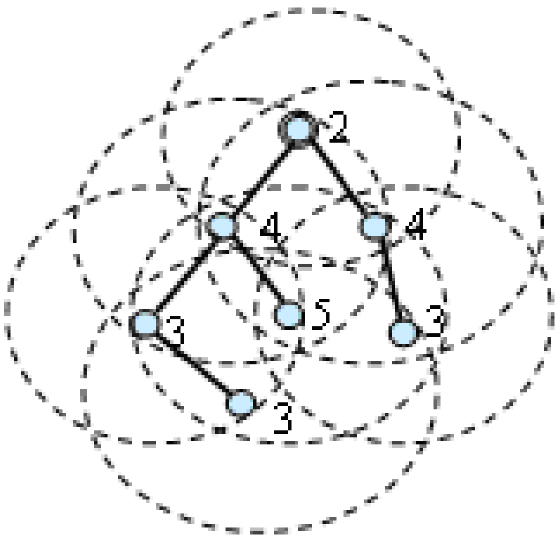


Figure 3.6: A tree with 7 nodes. Each node is labeled with the interference value. The intra-tree interference value of the tree is 4

Now, we can define the partition and channel assignment problem. Given k available orthogonal channels, the problem is to Partition a sensor network into k vertex-disjoint trees with Minimizing the maximum intra-tree Interference value of all Trees, called the PMIT problem. Next, we study its complexity.

3.2.2.2 The Complexity of the PMIT problem

The PMIT problem is similar to graph coloring problem, but different at that this problem requires that nodes with the same color construct a tree with minimum interference, while the graph coloring problem seeks independent sets for each color. Unfortunately, we find that this problem is also NP-complete, like the graph coloring problem. First, we restate this optimization problem as a decision problem. Given an integer d, the decision PMIT problem $< G,k,d >$ is to determine whether a graph G can be partitioned into k node-disjoint trees and the interference value of every tree is no more than d. The following theorem shows that this problem is NP-complete.

Theorem 3.1. *The PMIT problem is NP-Complete.*

Proof. First, it is clear that PMIT belongs to NP problem, because given a partition we can calculate the interference value of each non-leaf node of trees, and get the interference value of each tree. This verification can be performed straightforwardly in polynomial time.

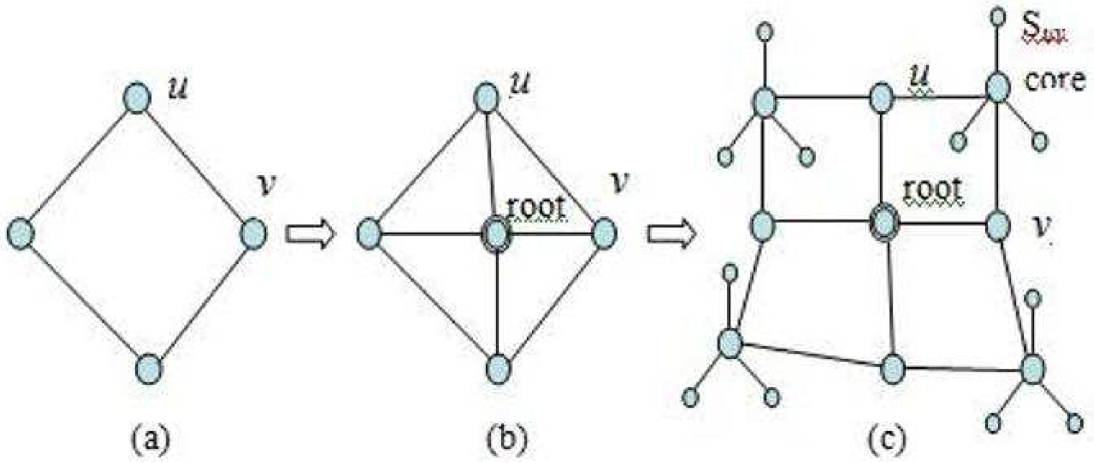


Figure 3.7: Reducing k-coloring to PMIT. (a) original graph. (b) after adding a root. (c) after adding stars.

Next, we prove that the PMIT problem is NP-hard by reducing the k-coloring problem to PMIT. Given a graph $G = (V,E)$ and k colors, the k-coloring problem is to determine whether each node can be assigned one color such that adjacent nodes must have different colors. Before the reduction, we first calculate the maximum degree Δ among all vertices of G. Then we define a structure $\Delta + 1 - star$, where a $\Delta + 1$-star consists of a vertex as the core and $\Delta + 1$ other vertices which are all adjacent to the core but have no edges to each other. The reduction algorithm takes as input an instance $< G, k >$ of k-coloring problem. It modifies the graph G into a new graph $G' = (V', E')$ as follows. First, we add a new vertex r as the root, and directly connect the root to every vertex in G. Then, for each edge $(u, v) \in E$, delete this edge, add a new $\Delta + 1$-star, called S_{uv}, into the graph and connects u and v to the core of this star respectively. At last, we calculate the proper interference range of each vertex, such that the interference disk of one vertex only covers its neighbors; in other words, the interference number of a vertex equals to its degree. After these operations, we get a new instance of the decision PMIT problem $< G', k, \Delta + 2 >$. Obviously, this reduction can be done in polynomial time. Figure 3.7 illustrates an example of the reduction. In the original graph, $\Delta + 1 = 2$. We first add the root, and then add 3-stars to replace every edge of the original graph.

Now, we show that the graph G can be k-colored if and only if the PMIT problem $< G', k, \Delta + 2 >$ can be satisfied. First, suppose that graph G' can be partitioned into a set G of trees T_1, T_2, $\ldots$, T_k, which satisfy PMIT problem. Then we compute a collection Γ of sets of vertices $C_1, C_2, \ldots, C_k$, where $C_i = T_i \cap V$. We claim that Γ is a proper k coloring for graph G. First, it is clear that every vertex of G is contained in one set of Γ. Secondly, for an arbitrary edge (u, v) in G, if both u and v are in the same tree T after partitioning G', the star S_{uv} must also belong to T, because the star only connects to u and v. In T, the core of S_{uv} is not a leaf and the interference value of the core is $\Delta + 3$, since the core has $\Delta + 3$ neighbors. Then the interference value of T must be at least $\Delta + 3$, which contradicts the constraint of the PMIT problem $< G', k, \Delta + 2 >$. So, we prove that if any two vertices u and v are adjacent in G, they are not in the same tree of G, which means that they are not in the same set of Γ. Therefore, Γ is a proper k coloring for graph G.

On the other hand, if there is a proper k coloring Λ for graph G, where $\Lambda = \{C_1, C_2, \ldots, C_k\}$, we can construct a set of trees in G' satisfying the PMIT problem $< G', k, \Delta + 2 >$. First, we add the root r into every set C_i of Λ, and then for every star S_{uv}, because u and v are adjacent, they must be in two different sets C_u and C_v. We arbitrarily put the star S_{uv} into one of two sets. Suppose it is the set C_u, and then C_u induces a tree in G', in which original vertex u of G connects directly to the root r, and the star S_{uv} connects to u. Next, consider the interference value of T. Since the maximum degree of u is Δ in G, then there are at most Δ stars connecting to u. The interference number of u is at most $\Delta+1$, plus the root. For the core of the star S_{uv}, the interference number is $\Delta+1$, because it lies in a $\Delta+1$-star, and only u is in the same tree. Thus the interference value of T is at most $\Delta+2$. Therefore, we can finally find a set of trees satisfying PMIT.

Above all, we prove that the reduction is legal. Since k-coloring problem is NP-hard, the PMIT is also NP-hard.

■

In the light of NP-completeness, there is no polynomial time exact algorithm which can always find the optimal partition. In next subsection, we introduce a greedy heuristic for the PMIT problem.

3.2.2.3 The PMIT Algorithm

In this algorithm, we assume that the interference sets of all nodes are already known. For a node u, let c_u denote u's channel, and p_u denote u's parent.

Algorithm 1 Greedy PMIT

```
Input: $k$ channels, a graph $G = (V,E)$, a root $r$ and the interference set of every node.
Output: For each node $u$, $c_u$ and $p_u$
  Use BFS-Fat-tree algorithm to construct a fat-tree with rooted at r.
  for each channel $i$ do
    $T_i = r$;
  end for
  for each node $u$ do
    $c_u = 0$; $p_u = null$;
  end for
  $level = 1$;
  repeat
    $node\_list = \{u|height(u) == level; c_u == 0\}$
    sort $node\_list$ in ascending order by the number of node's parents.
    for each node $u$ in $node\_list$ do
      find $T_i$ which keep connected and has the least interference after adding $u$.
      $T_i = T_i \cup \{u\}$; $c_u = i$; $p_u = v$, which connects to $u$ and has the least interference among all
      nodes in $T_i$.
      update the interference value of $T_i$.
    end for
    $level++$;
  until $level >$ the maximum height of the fat tree
```

Algorithm 2 Breadth-First-Search Fat tree

```
Input: a graph $G = (V,E)$ and a root $r$.
Output: For every node $u$, its parent set $parent(u)$ and its height in the tree $height(u)$.
  for each node $u$ in $G$ do
    $height(u) = max - Integer$; $parent(u) = null$;
  end for
  $S = r$; $height(r) = 0$;
  for each node $u$ in $S$ do
    for each node $u$'s neighbor $v$ do
      if $height(v) > height(u)$ then
        $height(v) = height(u) + 1$;
        $parent(v) = parent(v) \cup \{u\}$;
        $S = S \cup \{v\}$;
      end if
    end for
  end for
```

This algorithm firstly applies a Breadth-First search algorithm to compute a fat tree rooted at the base station. There are two important properties of the fat tree. First, nodes keep its height and have multiple parents on the fat tree. Secondly, the fat tree is actually a shortest path tree, where branches from the base station to each node are paths with the least hop count, because we use BFS strategy to build the tree.

Next, we execute the channel allocation one-by-one level from top to bottom on the fat tree. At each level, we always process nodes with fewer parents first, because they are less free to choose channels. For each node, we choose an optimal channel, in other words select an optimal tree to add the node in. The criterion is that the tree must connect to the node, and adding the node brings the least interference to this tree. If multiple trees tie, the tree with fewer nodes is chosen. After a node joining a tree, it selects a parent which has the least interference value among all possible parents within the tree selected. It is clear that the algorithm covers all nodes of graphs, and when a node gets a channel, the algorithm ensures it connects to one tree rooted at r, which demonstrates the correctness of the algorithm. The following theorem states the time complexity of the algorithm.

Theorem 3.2. *The time complexity of the Greedy PMIT algorithm is $O(d \times k \times n^2)$, where d is the diameter of the graph, n is the number of nodes, and k is the number of channels.*

Proof. The time complexity of construing a FAT tree is $O(d \times \Delta \times n)$, where Δ is the maximum degree in the graph. In PMIT algorithm, Step 12 takes $O(k \times n)$ in the worst case, and the loop beginning at Step 11 may run at most n time. Thus, the procedure within the repeat loop takes $O(k \times n^2)$, and the repeat loop iterates at most d times, because the tree height never exceeds the diameter of the graphs. The time complexity is $O(d \times k \times n^2)$ in the worst case. ■

A good property of this algorithm is that every node keeps the shortest path to the base-station. This property comes from the fact that the algorithm processes nodes one-by-one level from top to bottom of this fat tree. Therefore, this partitioning algorithm does not require extra transmissions and does not reduce the end-to-end delivery ratios, neither increase energy consumption during data collection.

This algorithm can be easily modified to a distributed algorithm because it only needs a local search at each node. First, nodes can construct a fat tree by broadcasting messages. During channel allocation, nodes make their own decision based on message from their parents, and notify their children. Also, since the network is static, we can run the centralized algorithm once at the beginning, or very infrequently, which is still practical even for large WSNs.

3.2.2.4 Evaluation of the Greedy Algorithm

As mentioned earlier, the network partition and channel assignment are very crucial to network performance improvement. In this subsection, we evaluate the performance of our greedy algorithm. We develop a graph simulator in JAVA, which can randomly generate a graph, and apply different schemes to do the channel allocation. In all experiments, we simulate a $200m \times 200m$ field, 250 nodes are uniformly distributed in the field, and the communication range is 10∼35m and interference range is always 1.5 times as the communication range. Since we are the first to study the PMIT problem, there are no other PMIT algorithms we can compare against. We use three alternative schemes as comparisons. One is to apply the Prim's algorithm to construct a minimum spanning tree as the data collection tree. This scheme is referred to as a base scheme with a single channel. Secondly, we implement the Eavesdropping channel assignment method proposed in [118]. We refer to it as a typical method used by node-based protocols. Note that this scheme does not ensure the connectivity among nodes in each channel. Next, we find the maximum interference value ρ among all nodes, and use ρ/k as the lower bound of the interference value after allocating k channels. Finally, we run the greedy algorithm and measure the maximum interference value among all trees after partitioning. In all experiments, each data point comes from the average result of 50 repeated experiments. For each data point, we also give its 90% confidence interval.

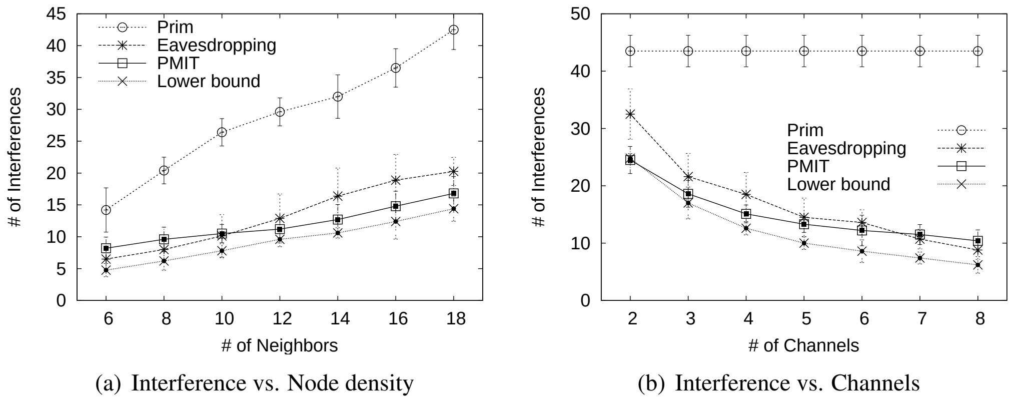


Figure 3.8: Performance Evaluation of the PMIT algorithm

In the first set of experiment, we use 3 channels and vary the number of neighbors by adjusting the communication range. The result is shown in Figure 3.8(a). We can see that the greedy algorithm can always get around 1/3 interferences of the Prim's algorithm with a single channel, which shows that our algorithm efficiently utilize 3 channels to decrease interferences. Comparing with Eavesdropping algorithm, we see that when the density is low, the Eavesdropping has less interferences than ours, mainly because it does not ensure the connectivity, but when the density becomes larger, the greedy algorithm outperforms the random scheme, for example when the density is 18, it gets 17% less interferences than Eavesdropping scheme. The reason is two-fold. First, when the density is large, there are no enough channels for nodes in two hop neighbors, so Eavesdropping have to randomly choose channels among nodes, which makes the maximum interference relatively large. But our algorithm always tries to find the local optimal, which can achieve more stable performance. Secondly, when the density is large, our greedy algorithm has more chances to set nodes with large interferences as leaves, which can further reduce interferences of subtrees. Finally, when comparing with the lower bound, the result of our algorithm is close to the lower bound of the interference value, and more importantly, the gap does not scale up with the density increasing, which suggests that our greedy algorithm has a good scalability with different densities.

In the second experiment, the radio range is 35m and we change the number of available channels. Results are illustrated in Figure 3.8(b). It is clear that with the small number of channels, our PMIT algorithm compute less interferences than Eavesdropping scheme, especially, when only 2 channels can be used, our algorithm has 24% less interferences than Eavesdropping scheme, and 51% less than a single channel. With more channels, performances of two schemes become closer. When there are 8 channels, Eavesdropping scheme has 18% less interference than our greedy algorithms. Comparing with the lower bound, we can see that with the small number of channels, our algorithm computes almost the same number of interferences as the lower bound.

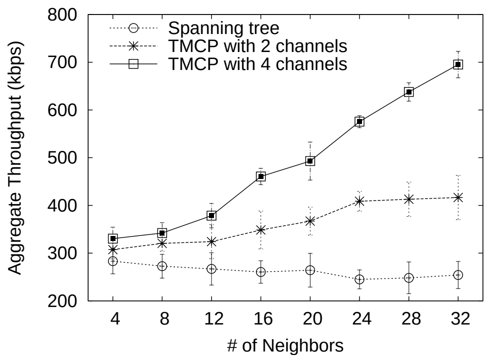
3.2.3 Performance Evaluation of TMCP

TMCP uses the greedy algorithm in the channel assignment component. In this section, we evaluate the communication performance of TMCP, by simulation and by experiments in a real testbed.

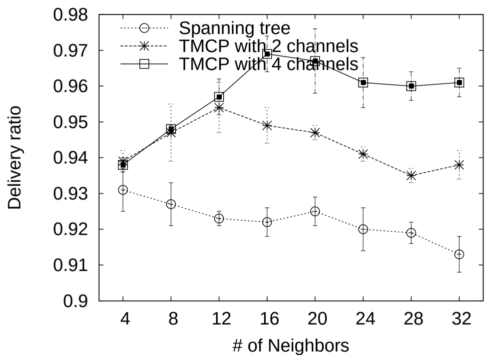
3.2.3.1 Simulation Evaluation

First, we evaluate the performance of TMCP through simulation experiments. We implement TMCP in GloMoSim. We use the same setting as simulations in section 3.2.2.4, where the communication range is 10$\sim$40m and the interference range is always 1.5 times as the communication range. This communication model is typically used to simulate the RF model of the CC2420 radio. Also, in the MAC layer, we use CSMA with the ACK-retransmission mechanism, which ensures that most packets can be received.

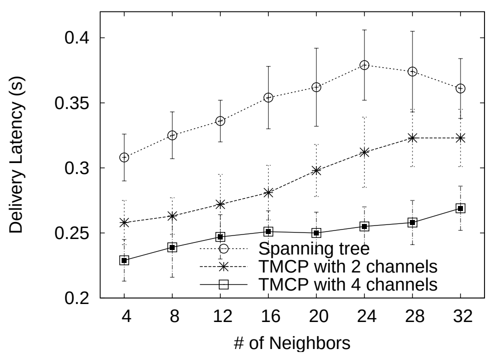
We conduct three sets of experiments. In the first two experiments, we compare TMCP with 2 and 4 channels and a spanning tree routing protocol with a single channel. First, we measure network performance with different node density. In this experiment, there are 50 Many-to-one CBR streams in the network, and the rate of each CBR is 40 packets per second. Results are shown in Figure 3.9, with the 90% confidence interval of each data point. According to the results, TMCP outperforms the original protocol in the following aspects. 1) By using the sophisticated network partition and frequency assignment algorithm, TMCP with 2 and 4 channels can decrease potential transmission collisions, which leads to an average 1.6 and 2.7 times higher aggregate throughput than the spanning tree algorithm. 2) By splitting traffic into different subtrees, TMCP decreases radio collisions as well as traffic congestion, which leads to higher packet delivery ratios and lower latency. 3) When the node density is increasing, TMCP shows good scalability. For example, in Figure 3.9(a) TMCP with 4 channels results in an increasing throughput as the number of neighbors increases, because with more nodes, TMCP more evenly partitions and channel allocation, which leads to better spatial reuse of concurrent transmissions. TMCP with 2 channels also shows this trend, but stops increasing the throughput when nodes have more than 20 neighbors, because the number of interferences exceeds the capacity of 2 channels.



(a) Throughput vs. Node density



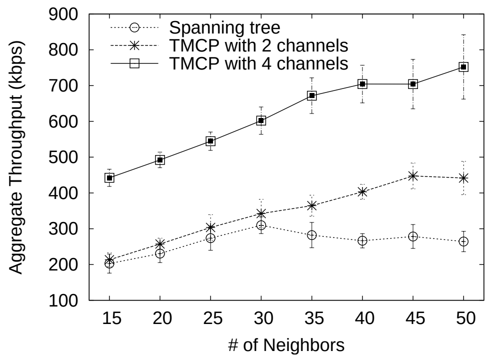
(b) Delivery ratio vs. Node density



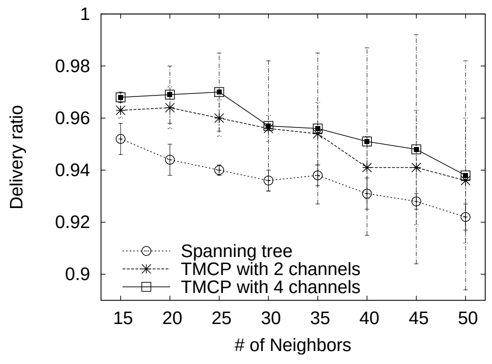
(c) Delivery latency vs. Node density

Figure 3.9: Performance with different node density

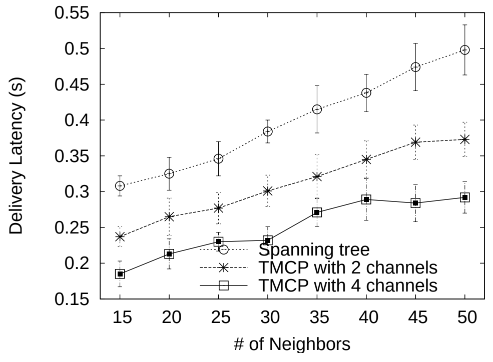
Second, we measure the performance with different network workloads. In Figure 3.10, we



(a) Throughput vs. # CBR streams



(b) Delivery ratio vs. # CBR streams

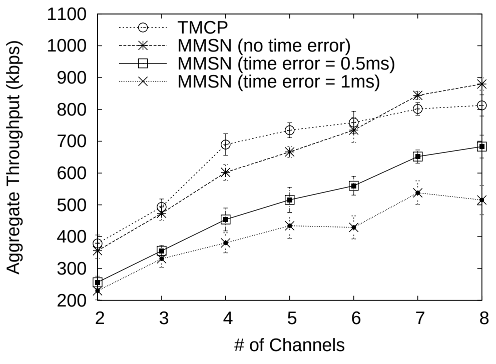


(c) Delivery latency vs. # CBR streams

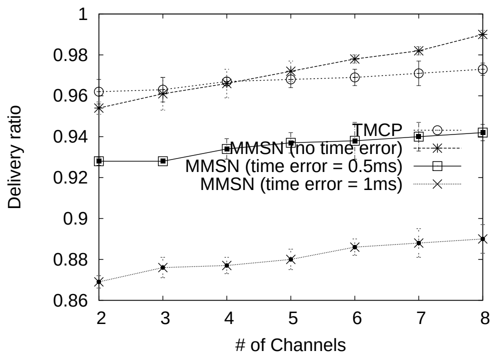
Figure 3.10: Performance with different network workloads

see that TMCP always exhibits better performance than the spanning tree protocol, especially in heavy workloads. For example, with 50 CBR streams TMCP with 4 channels achieves 2.8 times aggregated throughput and 42% lower delivery latency than the spanning tree. Also, the spanning tree protocol has a decreased packet delivery ratio from 95.2% to 92.1% in Figure 3.10(b), while TMCP has a much smaller decrease. This is because TMCP splits heavy workloads into different trees and is more tolerant to system load variation than the spanning tree algorithm. However, we also find the performance of TMCP is unstable. For example, in Figure 3.10(b), when the workload increases, the variation of delivery ratios of TMCP becomes larger. This is because these CBR streams are not evenly distributed among subtrees, and flow congestion can occurs on subtrees at which many CBR streams cluster.

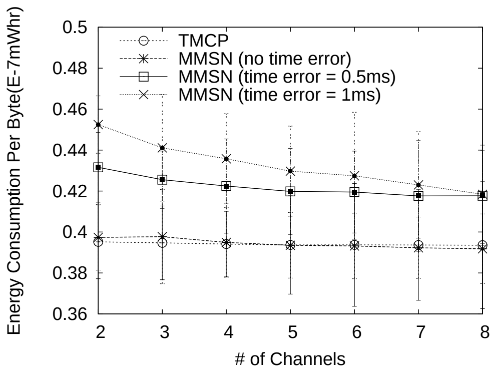
Last, we compare TMCP with MMSN [118], a typical node-based multi-channel protocol. In



(a) Throughput



(b) Delivery ratio



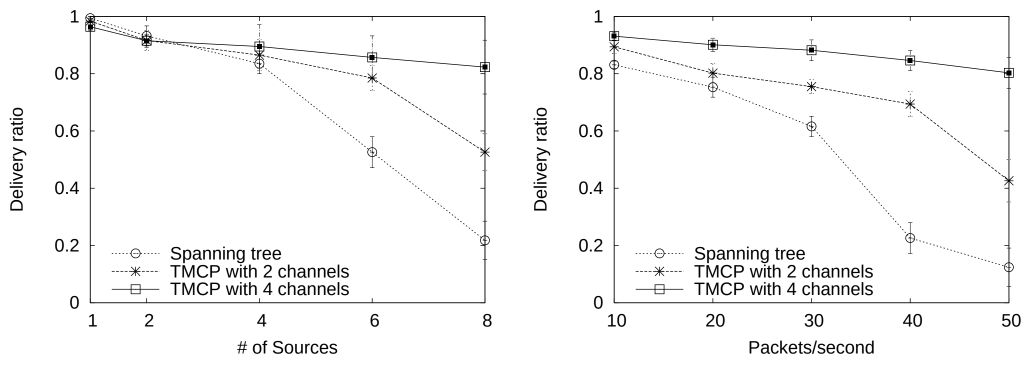
(c) Energy Consumption

Figure 3.11: Performance comparison of TMCP and MMSN

this group of experiments, 50 CBR streams are used and the node density is set to 38, by configuring the radio range to 40m. As mentioned in section 3.1, time synchronization errors may impact the performance of multi-channel protocols. Here with the number of channels changing, we compare TMCP and MMSN with different time errors. All results are presented in Figure 3.11. Here, we compare throughput, delivery ratio and energy consumption. Overall, the performance of TMCP and MMSN is very close. More precisely, when the number of channels is small, TMCP has a little better performance than MMSN. For example, in Figure 3.11(a), TMCP achieves a 10% higher throughput on average than MMSN with less than 5 channels. But when the number of channels increases, MMSN outperforms TMCP. This agrees with the evaluation results in section 3.2.2.4, where our channel assignment algorithm works better than other channel assignment schemes with a small number of channels. Also Figure 3.11(c) shows that the power consumption of TMCP and

MMSN are close. However, here we only consider the power consumption of data communication. As discussed in section 3.2.2.3, the channel assignment is executed infrequently, and that power consumption can be amortized during the time. On the other hand, time synchronization errors cause a great performance degradation for MMSN, but without any impact on TMCP. Considering multi-channel realities, TMCP is more suitable for practical WSNs than node-based multi-channel protocols.

3.2.3.2 Evaluation in a Real Testbed



(a) Delivery ratio vs. Number of Sources (b) Delivery ratio vs. Data rate

Figure 3.12: Evaluation in a test bed

Besides simulation evaluations, we also implement TMCP in a real testbed with Micaz motes. The testbed consisted of 20 Micaz motes, and four motes are laid closely together to act as a base station with four transceivers. Before the experiment, we first use the channel detection technique described in subsection 3.2.1 to find available orthogonal channels, and then run the channel assignment algorithm on a PC. After computing the assignment, the results are sent out to all motes. During the experiments, some nodes are selected as sources to transmit packets to the base station. We conduct two sets of experiment, and compare a normal spanning tree protocol with a single channel and TMCP with 2 and 4 channels. All experiments are repeated several times and averaged.

In the first set of experiments, while changing the number of sources, we measure the packet reception ratios. Here, all sources send packets with the data rate of 20 packets per second. The results are shown in Figure 3.12(a). We see that when the number of sources is above 4, the spanning tree protocol has low reception ratios below 60%, while TMCP with 2 channels can get high reception ratio until there are 8 sources and TMCP with 4 channels always maintains a high reception ratio. Performance gains of TMCP come from the fact that it effectively reduces interferences and mitigates congestion at nodes.

In the second set of experiments, we use 4 sources in the networks, and change the data rate of the sources. We also measure packet reception ratios at the base station. The result is shown at Figure 3.12(b). We see that the saturated data rate (reception ratio above 80%) of the sources is 20 packets per seconds for the spanning tree protocol. For TMCP with 2 channels, the saturated rate is 30 packets per seconds, and TMCP with 4 channels can support 50 packets per second. These experiments show that TMCP works well in real sensor networks.

Above all, our evaluation results show that TMCP better accommodate multi-channel realities found in WSNs than other multi-channel protocols, and more suitable to provide reliable, low-latency and robust communication for high-rate and dense WSNs for mission-critical applications.

3.3 Pruning TMCP: Reliable Tree-based Multi-channel Protocol

In the last section, we present TMCP, a novel tree-based Multi-channel protocols, which significantly improves network performance, in terms of throughput, delivery ratio and latency. Especially, TMCP outperforms existing protocols in dense networks with high volume traffic, which makes it a better solution for communication requirements of mission-critical WSN applications. TMCP is designed with the assumption that all links have the same quality, and low-level MAC protocols provide reliable point-to-point transmissions. However, this assumption may not hold for many mission-critical applications, where links in networks exhibit the great heterogeneity in link qualities. In [64], Shan et al. observe different packet reception ratios of links and identify two types (stable and unstable) of links in an indoor WSN system. This quality diversity is caused by multi-path fading of signals and shadowing effects of humans and obstacles. In [88], Sexton et al. measure the link characteristics in several industrial facilities, which consist of rooms with lots of metal surfaces and rotating machinery. Their results show that links in this environment have a wide quality range and vary substantially in stabilities. From all these observations, it is clear that multi-channel protocol design should consider the link diversity in WSNs, and address potential delivery failures caused by poor links.

Unfortunately, the original TMCP cannot guarantee reliable end-to-end deliveries in the presence of the link diversity. TMCP builds routing sub-trees to minimizing interferences. It is possible that routing trees consist of low-quality links, which leads to the delivery failures. In this section, we focus on how to extend TMCP to minimize interferences, as well as to meet end-to-end reliability requirements.

3.3.1 Model and Reliable Channel Assignment Problem

We extend the basic network model in Section 3.2.2.1 with the link quality metric. For every link (edge) e, $p(e)$ stands for the probability of successful transmission for a single attempt on this link. In realities, this probability can be obtained by link estimation methods in [33] [102]. Also, the function $pdr(e,x)$ denotes the probability of successful transmission for x attempts (retransmissions), and we have $pdr(e,x) = 1 - [1 - p(e)]^(x)$. Next, we assume that there is only one base station in the network. For node u, $E2EPDR(u)$ stands for the probability of successful transmission from node u to the base station. Suppose that packets are relayed along a k-hop path, and then we have $E2EPDR(u) = \prod_{i=1}^{k} pdr(e_i,x)$, where e_i is the ith link along the path. Finally, we use RR to denote the minimum acceptable probability of successful transmission from nodes to the base station. Given a RR, the communication reliability requirement is: $\forall u \in G, E2EPDR(u) \geq RR$.

With this model, we extend the PMIT problem in Section 3.2.2.1 to a new reliable channel assignment problem. Given k available orthogonal channels and the reliability requirement RR, the problem is to Partition a sensor network into k vertex-disjoint trees, such that (1) the partition minimizes the maximum intra-tree Interference value of all Trees; (2) $\forall u \in G, E2EPDR(u) \geq RR$. We call this problem as RPMIT problem. As an extension of the PMIT problem, the RPMIT problem is also NP-Complete.

3.3.2 The pruning algorithm

As mentioned before, TMCP effectively reduces interferences by building optimal routing subtrees, but it cannot meet the reliability requirements, because subtrees may consist of low-quality links. In order to address this issue, we propose a two-step solution. First, we use a pruning algorithm to remove all links which cannot meet the reliability requirement, and then run TMCP on the remaining fat tree to minimize interferences.

In order to prune the fat tree, we firstly use a downward pruning. During the iteration, we first compute the $E2EPDR$ of parent nodes, and then remove links that connect to child nodes but make children's $E2EPDR$ lower than RR. Since a node may have multiple path to the root on the fat tree, one question is which $E2EPDR$ should be used to compute the $E2EPDR$ of its children. The first option is to use the minimum $E2EPDR$ which always guarantees that all remaining paths can meet the reliability requirement. But this over strict pruning removes links that actually can be used in a quality path, and may make it impossible for low-level nodes to find a path. It also makes the remaining fat tree too sparse, thus reduce the choices for the channel assignment. On the contrary, the second option is to use the maximum $E2EPDR$, which only removes links whose qualities are too poor and cannot be used in any path. But it cannot guarantee that all remaining paths are qualified.

We choose to use the maximum $E2EPDR$ during the downward pruning. After that, we further run an upward pruning from leaf nodes to the root. The upward pruning aims to remove low-quality links to guarantee that all remaining paths are qualified. We introduce a new node property, the required end-to-end delivery ratio, denoted as $RE2EPDR$. For a node u, $RE2EPDR(u)$ is defined as the minimum $E2EPDR$ that allows u's descendants to meet the reliability requirements. $RE2EPDR(u)$ can be computed by:

$$RE2EPDR(u) = \min_{v \in \{u' s children\}} \left(\frac{RR}{RE2EPDR(v) * pdr(e_{u,v}, x)} \right) \tag{3.2}$$

During the upward pruning, we compute the $RE2EPDR(u)$ for any node u, and then remove u's upstreaming link $e_{u,p}$, if $pdr(e_{u,p}, x) < RE2EPDR(u)$. Here, the upstreaming link e is the link connecting u and u's parent p. This upward pruning guarantees that all remaining paths meet the reliability requirement, because it only keeps links that satisfy all descendants' $RE2EPDR$ requirements. The detailed algorithm is presented in the following.

Our pruning algorithm has two good properties. First, it keeps the connectivity of the graph. More specifically, if there exist a qualified path connecting a node u to the root in the original graph, it is guaranteed that u still connects to the root with a qualified path after pruning. Second, the algorithm guarantees that the remaining network consists of only qualified path. This property results from the upward pruning which removes all links that cannot meet the minimum reliability requirement. The following theorem states the time complexity of the algorithm.

Theorem 3.3. *The time complexity of the pruning algorithm is $O(d \times \Delta \times n)$, where d is the diameter of the graph, Δ is the maximum degree in the graph and n is the number of nodes.*

Proof. The time complexity of the pruning algorithm is equal to the time complexity of construing a FAT tree, which is $O(d \times \Delta \times n)$ ■

As well as TMCP, this pruning algorithm can be changed to a distributed algorithm because every node only needs to collect information from its neighbors (parents and children). In real deployment, we can first measure link qualities in networks, and run the pruning algorithm and TMCP at the beginning. During run-time, the pruning algorithm and TMCP can be triggered periodically to ensure the communication reliability.

3.3.3 Evaluation of the Pruning TMCP

In this subsection, we evaluate the performance of the new TMCP in two steps. First, we analyze the performance of the original and new TMCP by observing their impact on two network properties. The first property is the percentage of reliable end-to-end routes, defined as the percentage of nodes which have a qualified route to the base station over all nodes. The latter metric is the number of potential interferences. We develop a graph simulator in JAVA, which can randomly generate a graph, and apply different schemes to do the channel allocation. In all experiments, we simulate a $200m \times 200m$ field, 250 nodes are uniformly distributed in the field, and the communication range

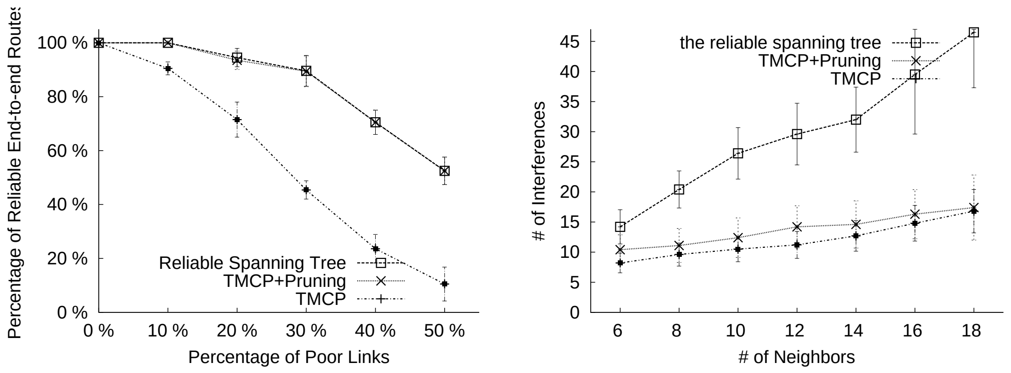
Algorithm 3 The Pruning Algorithm

```
Input: a graph $G = (V,E)$, a root $r$, a link quality profile $p(e)$ for every $e \in E$, the maximum
  number of retransmissions $n$ and the end-to-end reliability requirement $RR$.
Output: a new graph $G' = (V,E')$
  $G' \leftarrow G$
  Use BFS-Fat-tree algorithm to construct a fat-tree with rooted at r.
  $h \leftarrow 0$
  while $h <=$ the fat-tree's $max\_height$ do
    for each node $u$ in the tree level $h$ do
      $max\_E2EPDR \leftarrow 0$
      for each parent $p$ in $parent(u)$ do
        if $E2EPDR(p) * pdr(e_{u,p}, n) < RR$ then
          remove $e_{u,p}$ from $G'$
        else
          if $E2EPDR(p) * pdr(e_{u,p}, n) > max\_E2EPDR$ then
            $max\_E2EPDR \leftarrow E2EPDR(p) * pdr(e_{u,p}, n)$
          end if
        end if
      end for
      $E2EPDR(u) \leftarrow max\_E2EPDR$
    end for
    $h \leftarrow h + 1$
  end while
  for each leaf node $u$ do
    $RE2EPDR(u) \leftarrow 1$
    for each parent $p$ in $parent(u)$ do
      if $pdr(e_{u,p}, x) < RE2EPDR(u)$ then
        remove $e_{u,p}$ from $G'$
      end if
    end for
  end for
  $h \leftarrow$ the fat-tree's $max\_height - 1$
  while $h > 0$ do
    for each node $u$ in the tree level $h$ do
      $min\_RE2EPDR \leftarrow 1$
      for each child node $c$ in $children(u)$ do
        if $\frac{RR}{RE2EPDR(c)*pdr(e_{u,c},x)} < min\_RE2EPDR$ then
          $min\_RE2EPDR \leftarrow \frac{RR}{RE2EPDR(v)*pdr(e_{u,c},x)}$
        end if
      end for
      $RE2EPDR(u) \leftarrow min\_RE2EPDR$
      for each parent $p$ in $parent(u)$ do
        if $pdr(e_{u,p}, x) < RE2EPDR(u)$ then
          remove $e_{u,p}$ from $G'$
        end if
      end for
    end for
    $h \leftarrow h - 1$
  end while
```

is 10$\sim$35m and interference range is always 1.5 times as the communication range. In order to simulate the link diversity in real systems, we follow the link quality model in [64], where links are categorized as poor links and good links. In simulations, the packet reception ratio of one good link is randomly selected in the range of $[0.9, 1]$, while that of poor links in $[0.5, 0.8]$. We also assume that the maximum number of retransmission is 2, the end-to-end reliability requirement RR is 80%, and there are 3 available channels to use. Besides the original and Pruning TMCP, we also run the reliable spanning tree algorithm, which use Dijkstra's algorithm to compute a spanning tree on which each node connects to the base station through the maximum reliability (weight) route. This approach can always achieve the maximum percentage of reliable end-to-end routes, but cannot reduce the number of potential collisions. We refer it as the best single-channel tree-base routing scheme. In all experiments, each data point comes from the average result of 50 repeated experiments. For each data point, we also give its 90% confidence interval.

In the first set of experiments, we study the performance of all three approaches with different-level link diversities, by varying the ratio of poor links in networks. All results are shown in Figure 3.13(a). All three algorithms compute less reliable routes when poor links increase in networks, but the reliable spanning tree and the pruning TMCP compute much more reliable routes than TMCP. For instance, with 30% poor links, TMCP makes only 42% nodes with reliable routes, but the pruning TMCP makes over 85% nodes with reliable routes. This reliability improvement comes from the fact that the pruning algorithm removes poor links which lead to unreliable routes. Furthermore, it is clear that the Pruning TMCP has the almost same result with the reliable spanning tree scheme. As aforementioned, the reliable spanning tree can always find a reliable route if existed. This result demonstrates that the pruning algorithm prunes sufficient links and ensures that the remaining network consists of only reliable routes.

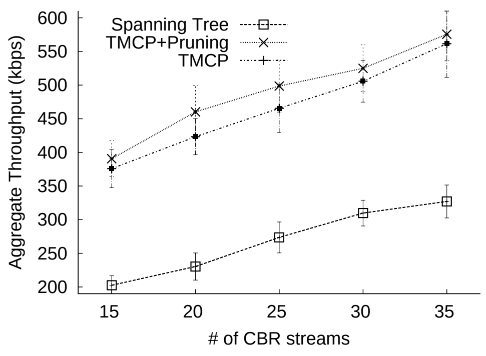
In the second set of experiments, we observe the performance of three approaches under different node densities, by varying the number of neighbors. In these experiments, networks consists of 20% poor links. As shown in Figure 3.13(b), both TMCP schemes get much less potential interferences of the reliable spanning tree scheme, which demonstrates that the channel assignment scheme efficiently utilizes 3 channels to reduce interferences. Comparing with the original TMCP, this new pruning TMCP has more interferences. For example, with 10 neighbors,the pruning TMCP gets around 12 potential interferences, while the original TMCP has around 10 interferences. This performance degradation is expected because the pruning algorithm removes a number of links, make the network sparser, and then reduce channel selection options. Overall,the pruning TMCP significantly improves the end-to-end reliability with the cost of slightly more potential interferences, which makes it a better solution for reliability-sensitive applications.



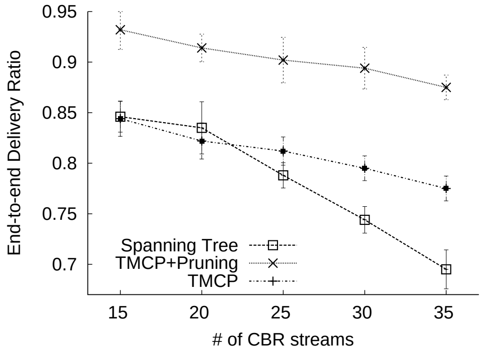
(a) Reliable end-to-end routes with different-level link diversity (b) Numbers of potential collisions with different densities

Figure 3.13: Performance Evaluation of the Pruning TMCP algorithm

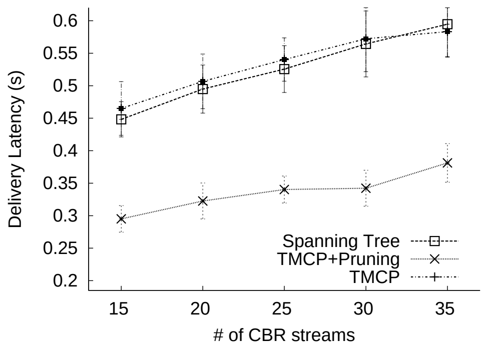
As the second step of our evaluation, we measure the general performance of the pruning TMCP with network traffic. The performance metrics include throughput, end-to-end delivery ratio, and latency. We implement both TMCP in GloMoSim. In this experiment, the percentage of poor links in networks is 20%, the average number of neighbors is 10, and the number of available channels is 4. Other settings are the same with previous experiments. Since mission-critical applications trigger burst and crowed traffic, our evaluation focuses on studying the performance with different network workloads, by varying the number of CBR streams in networks. Results are shown in In Figure 3.14. We can see that the Pruning TMCP always exhibits better performance than the spanning tree and the original TMCP, especially with higher reliability and lower latency. For example, with 35 CBR streams, the pruning TMCP improves end-to-end delivery ratio by 12% and reduces the latency by 35% over TMCP. This performance improvement is because the pruning TMCP removes poor links, and tends to choose better links to build the routing tree, which increase



(a) Throughput vs. # CBR streams



(b) Delivery ratio vs. # CBR streams



(c) Delivery latency vs. # CBR streams

Figure 3.14: Performance with different network workloads

the likelihood of successful transmissions and reduce the number of retransmissions. Furthermore, the pruning TMCP is more tolerant to workload increase than TMCP. When the number of CBR streams increases, all three schemes have a reliability decrease and a latency increase. But the pruning TMCP has the least performance degradation among all three. Finally, we also observe that the performance of the pruning TMCP is still unstable, indicated by the large confidence interval. More specifically, the pruning TMCP achieves poor performance, if CBR streams are not evenly distributed among routing trees, which causes the unbalanced load among channels and lead to more collisions and congestions. This observation indicates the necessity of taking the traffic information into channel assignment consideration.

3.4 Traffic Aware Channel Assignment

In previous sections, we present TMCP and the pruning TMCP. Like most other existing multi-channel protocols, these two schemes offer some static solutions to channel allocations, aiming to minimize potential interference among nodes. Since topologies of sensor networks are quite static, such static solutions can be executed in the deployment time, or infrequently during runtime, and they help MAC protocols improve communication performance on average.

In this section, we focus on channel assignment problems in sensor networks. We believe that existing static approaches for channel assignment are insufficient because of two reasons. First, they try to reduce potential interference with the assumption that all nodes have the same amount of traffic to carry simultaneously. This assumption is not true for most WSNs, where only a fraction of nodes need to transmit packets at any time. Second, even though a specific sensor network is deployed statically, the traffic volume and pattern can vary significantly both across the deployment area and across time. For example, a military intrusion detection sensor network [39] may have a regular and low speed traffic involving a few nodes when no intrusion is occurring, but may experience a large burst of traffic affecting a lot of nodes when enemy tanks are detected. Such traffic variability can change the interference pattern, and hence a multi-channel MAC with static channel assignment will severely suffer in terms of performance.

To improve current channel assignment solutions, we develop and systematically study the notion of traffic-aware channel assignment for WSNs. We start by considering a setting where perfect information about current and future traffic is available. Then we propose a new channel assignment scheme which exploits this traffic information to minimize interference occurring with real traffic. We compare this scheme with two typical static channel assignment schemes by simulation, and results show that being traffic-aware can substantially improve the performance of channel assignment. This baseline analysis helps establish the potential benefits of traffic-aware channel assignment algorithms.

3.4.1 Channel Assignment Scheme

In this subsection, we first explain two typical static channel assignment schemes [118]:even selection and eavesdropping, and then propose a new traffic-aware channel assignment approach, which uses traffic information to achieve load balance among channels and effectively reduces runtime system interference. Lastly, we compare our traffic-aware channel assignment with the two existing approaches through simulation evaluation.

3.4.1.1 Static Channel Assignment

In channel assignment, each node is assigned a physical channel for data reception. The assigned channel is broadcast to its neighbors, so that each node knows what channel to use to transmit unicast packets to each of its neighbors. In order to reduce communication interference and hence reduce hidden terminal problems [43], static solutions evenly assign available channels to nodes within two communication hops. In WSNs, such static channel assignments are considered to either be done once at the beginning of the system deployment, or it can be done very infrequently for adaptation to system aging. In this subsection, we describe two channel assignment schemes proposed in [118]: even selection and eavesdropping.

Even Selection In even selection assignment, nodes first exchange their IDs among two communication hops [117], so that each node knows its two-hop neighbors's IDs. To achieve this, each node first beacons its node ID to neighbors, so that each node knows its neighbors's IDs within one communication hop. Then, each node beacons again, broadcasting all neighbors's IDs it has collected during the previous beacon round. Therefore, after two rounds of beacons, all nodes get their neighbors' IDs within two communication hops.

After the two rounds of beacons, nodes begin to choose data receiving frequencies (or channels) in the increasing order of their ID values. If a node has the smallest ID among its two communication hops, it chooses a channel first and it chooses the smallest channel among available channels. The node then beacons its channel choice within two hops. If a node's ID is not the smallest among two hops, it waits for channel decisions from other nodes within two hops that have smaller IDs. When decisions from all those nodes are made and are also received, the node chooses the smallest available (not chosen by any of its two-hop neighbors) channel. If all channels have been chosen by at least one two-hop neighbor, it randomly chooses one of the least chosen channels. After picking a channel, the node broadcasts its choice within two hops. We call this scheme even selection, which makes an even allocation of available frequencies to all nodes within any two communication hops. When the number of frequencies is at least as large as the two-hop node number, even selection guarantees to assign different frequencies to different nodes within any two-hop neighborhood. When the number of frequencies is small, even selection allows two-hop neighbors evenly share the available frequencies.

Eavesdropping We observe that although even selection results in even sharing of available frequencies among two-hop neighbors, it requires a number of two-hop broadcasts. To reduce the communication overhead, a lightweight eavesdropping scheme is also proposed in [118]. In eavesdropping, each node takes a random backoff before it broadcasts its physical channel decision. Each node also records overheard physical channel choices during the backoff period. After the random backoff, a node randomly chooses one of the least chosen frequencies for data reception. Compared with even selection, eavesdropping requires less communication overhead, but leads to more potential conflicts since it only collects information within one hop rather than two hops for channel decisions.

3.4.1.2 Traffic-Aware Channel Assignment

In this section, we introduce a traffic-aware channel assignment scheme. Here, the term ”traffic aware” means that nodes have knowledge about future traffic. More precisely, nodes know their reception data rates of the future. Now, we assume that the traffic data rate does not change, while in the future we intend to discuss dynamic traffic. One practical problem is the dissemination of dynamic traffic information to nodes. One practical way is that at regular intervals, nodes calculate the reception data rate, and use it to determine channel assignment. Also, considering that sensor networks are used to periodically collect environment data in most scenarios, upper layers can pass such application information to the channel assignment component, and then the reception data rate can be inferred from those collection tasks’ settings.

Now, every node is assigned a traffic weight, which corresponds to its future reception data rate. The problem is to assign the right channel to each node, aiming to minimize the maximum load of any channels within the two-hop neighborhood of any node. Here, we choose this goal because the more load one channel takes, the more likely radio collisions occur in this channel. Also, the channel load affects the throughput and the latency of communication. We also find that this problem is very similar to the load balancing job scheduling problem, where each channel can be viewed as one machine, and the traffic weight of each node corresponds to the processing time of each job. The difference between these two problems is that in our channel assignment, we require the load balance within any two-hop neighborhood, but the job scheduling problem only asks the load balance for one group of machines. If the diameter of this network is two hops, our traffic-aware channel assignment problem is the exact same problem with the load balancing job scheduling problem. Since the job scheduling problem is NP-hard, it is clear that our traffic-aware channel assignment problem is also NP-hard.

In the light of NP-completeness, there is no polynomial time exact algorithm which can always find the optimal assignment. Next, we propose a greedy traffic-aware channel assignment scheme.

First, nodes exchange their IDs and their traffic weights among two communication hops, so that each node knows its two-hop neighbors' IDs and traffic weights. After nodes collect traffic information of all neighbors within two hops, they make channel decisions in the decreasing order of their traffic weight, with the smallest node ID used as a tie breaker. If a node has the greatest traffic weight among its two communication hops, it chooses the channel with the least load among available channels, and then beacons the channel choice within two hops. After receiving this beacon, nodes update the load of the corresponding channel. If a node's traffic weight is not the greatest one among two hops, it waits for channel decisions from other nodes within two hops that have greater weight. A node also waits for all nodes with equal weight but lower node ID to make decisions first. After decisions from all nodes with greater weight or equal weight but lower node ID are received, a node chooses the channel with the least load. Since nodes choose channels in sequence by decreasing weight with a node ID tiebreaker, our assignment algorithm will always converge for any set of nodes and traffic weights.

Table 3.1: Simulation Configuration

TERRAIN	(200m$\times$200m) Square
Node Number	289
Node Placement	Uniform
Application	Many-to-Many/Gossip CBR Streams
Payload Size	32 Bytes
Routing Layer	GF
MAC Layer	CSMA/MMSN
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250 Kbps
Radio Range	20m$\sim$45m

3.4.2 Performance Evaluation

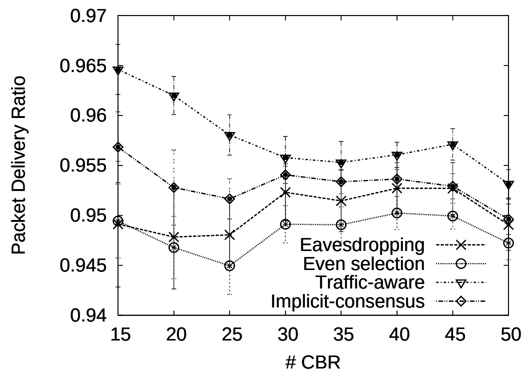
In this subsection, we compare the performance of two static channel assignment schemes and the new traffic-aware assignment. We assume that every node has perfect knowledge about its future reception data rate.

For this performance evaluation, two groups of experiments are designed. In the first group, different system loads are used, and in the second group of experiments, the number of available channels is varied.

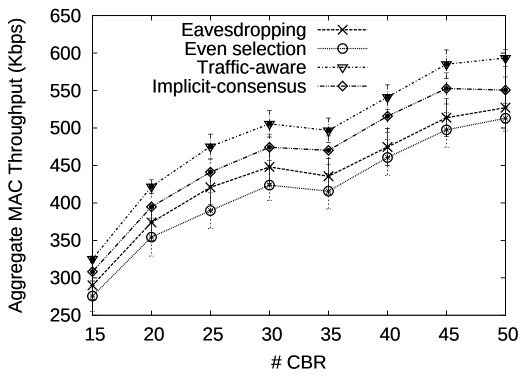
For all the two groups of experiments, four performance metrics are adopted: aggregate MAC throughput, packet delivery ratio, channel access delay, and energy consumption per successfully delivered data byte. The packet delivery ratio is calculated as the ratio of the total number of data packets successfully delivered by the MAC layer over the total number of data packets the network layer requests the MAC to transmit. The aggregate MAC throughput measures the performance gain and is calculated as the total amount of useful data successfully delivered through the MAC layer in the system per unit time. The channel access delay measures the time delay a data packet from the network layer waits for the channel before it gets sent out. The energy consumption per byte is the system wide energy consumed for successfully delivering one byte of user data.

During all the experiments, the Geographic Forwarding (GF) [51] routing protocol is used and the simulation is configured according to the settings in Table r̃eftab:glomo-config. For each data value we present in the results, we also give its 90% confidence interval.

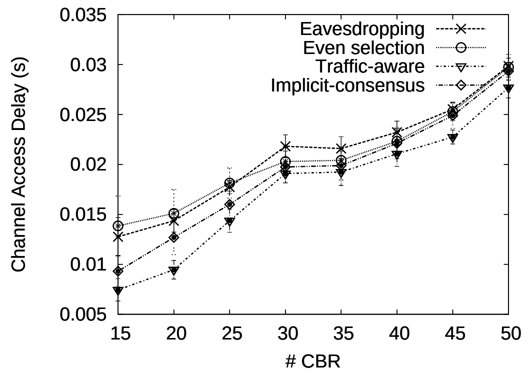
3.4.2.1 Performance Evaluation with Different System Loads



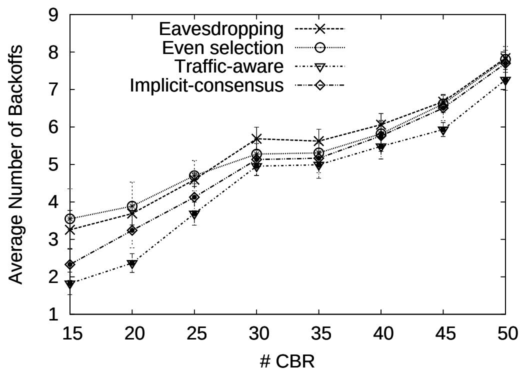
(a) Packet Delivery Ratio in MAC



(b) Aggregate Throughput in MAC



(c) Average Channel Access Delay



(d) Energy Consumption Per Delivered Data Byte

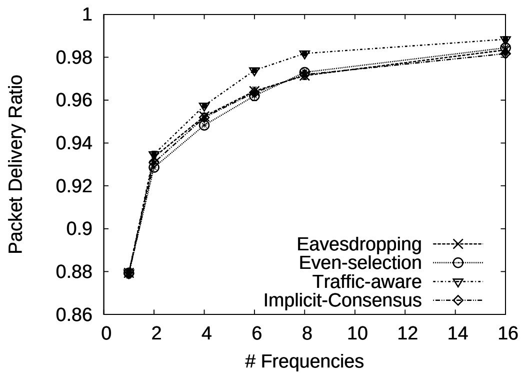
Figure 3.15: Performance Evaluation with Different System Loads

In the first group of experiments, we explore traffic-aware assignment's performance when different system loads are used, which are generated by different numbers of CBR streams. In the experiments, the node density is set to 38, and the number of available channels is 5.

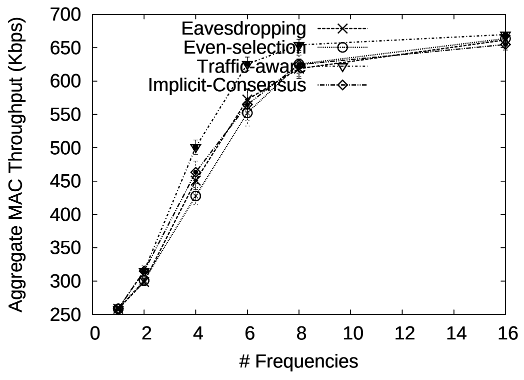
As Figure 3.15 shows, for all the system loads we configure from 15 CBR streams to 50 CBR streams, it is observed that traffic-aware assignment always exhibits better performance than static schemes in all performance metrics in all scenarios. For example, as shown in Figure 3.15, comparing with the best static scheme, traffic-aware scheme achieve average 13.5% higher aggregate throughput as shown in (b), and average 0.74 less backoff per packet shown in (d). It is clear that traffic-aware channel assignment effectively reduces the radio interference, and by keeping the load balance among channels, it mitigates packet congestions within channels, and leads to high throughput and lower latency.

Another interesting trend is that when the system load is light or heavy, the traffic-aware assignment outperforms static schemes with a large gap, but when the system load is medium, like 30 streams, the performance of the traffic-aware scheme is very close to static schemes. One possible reason is that with such medium load, most nodes have similar traffic weights, which make static schemes also get good performance.

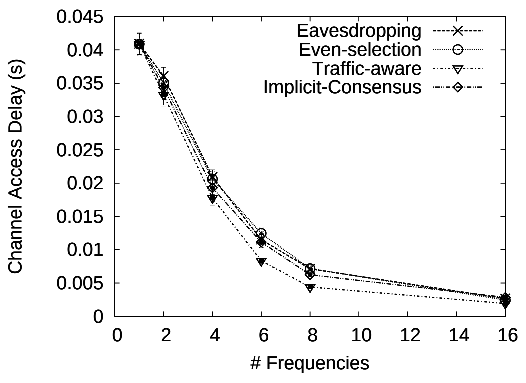
3.4.2.2 Performance Evaluation with Different numbers of channels



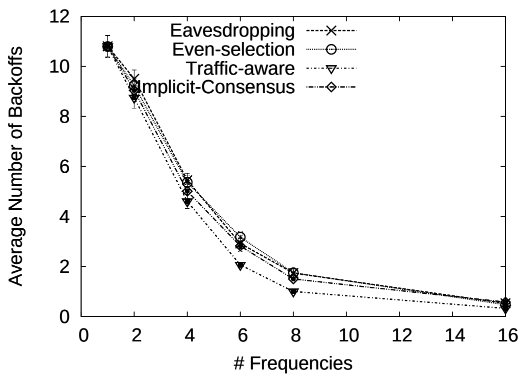
(a) Packet Delivery Ratio in MAC



(b) Aggregate Throughput in MAC



(c) Average Channel Access Delay



(d) Energy Consumption Per Delivered Data Byte

Figure 3.16: Performance Evaluation with Different numbers of channels

In many deployed sensor network systems, the number of available channels may vary. For example, in an indoor scenario, the interference from WiFi network may decrease the number of available channels that sensor networks can use. So, in the second group of experiments, we evaluate the performance of channel assignment schemes when different numbers of channels are utilized. The number is increased from 1 to 16, and a many-to-many traffic pattern is used that consists of 50 CBR streams.

Once again, the experimental results confirm that traffic-aware assignment always achieves a higher performance than static schemes, which can be observed in Figure 3.16 (a)$\sim$(d). The corresponding reasons can be found in the first groups of experiments and are not repeated here.

It is shown that when the channel number is small, for example 1 or 2, the performance of all four is very close. When we have many channels like 16, the performance is also close. On the other side, when the channel number is medium, like 4, 6, 8, the traffic-aware scheme obviously outperforms others. We believe that in practices, one sensor network may co-exist with other sensor networks or WiFi networks, and in most cases, the number of available channels is around such medium numbers.

Overall, our simulation evaluation demonstrates that the traffic-aware channel assignment greatly improves multi-channel MAC performance: it significantly enhances the the packet delivery ratio and throughput, while at the same time reducing channel access delay and energy consumption. This traffic-aware work is still in its early stage. In the future, we are going to study how to efficiently deal with the traffic variability during system runtime. Some questions are: how to predict the future traffic? When traffic varies and the prediction fails, how can we change channel assignment dynamically? Of course, solutions of these questions must not bring too much overhead, and must converge to a stable assignment in limited time.

3.5 Conclusions

This chapter focuses on the reliable and efficient delivery of large amounts of data in bandwidth-limited wireless sensor networks by making use of the multi-channel capability of the sensor radios and by using optimal routing topologies. We start with experimenting the operation of the sensor radios to characterize the behavior of multi-channel communication and identify the major challenges of design multi-channel protocols in WSNs. This reality provides important guidelines for multi-channel design in WSNs, and can benefit developing more sophisticated multi-frequency parallel communication for sensor networks in the future. We propose a set of algorithms to optimize the routing topologies and channel assignments, in order to improve delivery reliability and reduce latency. Our proposed protocols resolve the practical challenges in real systems, and are designed to provide high throughput and high delivery ratio during high-rate traffic whereas it also meets the traditional requirements of wireless sensor networks such as energy efficiency and scalability. Furthermore, some extended schemes are proposed to maintain performance in networks with high link diversity, as well as traffic diversities.

One limitation of our work is that current approach lacks the generality and flexibility to support diverse application requirements. For example, the pruning TMCP is suitable for networks with great link diversity, while traffic-aware channel assignment scheme is designed for networks with diverse traffic. It is interesting and necessary to find a consistent framework to integrate different multi-channel solution into the protocol stack and the WSN architecture, and support multiple topology, network and traffic patterns and QoS levels. We leave this as future work.

Chapter 4

Active Collision Recovery in Wireless Sensor Network

4.1 Introduction

Packet Collision occurs when two or more close stations attempt to transmit a packet at the same time, which results in packet loss and impedes network performance. Many CSMA based MAC protocols are proposed in Wireless Sensor Network (WSNs) to avoid collisions, such as B-MAC [80], X-MAC [15]. These protocols can efficiently reduce collisions, but intrinsically cannot eliminate all collisions, because of hidden terminal problems, as well as collisions when multiple nodes sense the medium free at the same time. Previous work [47] demonstrates that CSMA based protocols cannot avoid such collisions when the number of concurrent transmissions grows. Such collisions become severe in dense WSNs for two reasons. First, many dense sensor networks are event-driven and generate bursty spatially-correlated traffic, where multiple sensors in the same neighborhood all have messages to send simultaneously in response to the same event. Second, WSNs are typically equipped with few sinks to which packets from sensors converge. Such convergence causes many collisions around sinks, described as the "funneling effect" [3]. Furthermore, the consequences of packet collisions are serious to WSNs. Collisions can cause the loss of critical control information from base stations, and applications may fail.

Another way to avoid collisions is to apply multiple frequencies to eliminate transmission interferences, which is the topic of Chapter 3. However, our previous results demonstrate that multi-channel protocols cannot remove all collisions in dense WSNs, due to insufficient channels and channel coordination costs. When collisions still occur, a packet recovery method should be applied to recover lost packets and ensure end-to-end delivery. Traditional Automatic Repeat Request (ARQ) method uses acknowledgment and retransmissions for packet recovery. However, ARQ brings high latency due to waiting for ACKs, and retransmissions cause more collisions, especially with heavy traffic. New packet recovery methods exploit the Partial Corrupted Pattern (PCP), where only some bits are actually corrupted in a "lost" packet. For instance, Forward Error Correction (FEC) helps receivers recover such partially corrupted packets by sending packets with redundant bits. As shown in [4], it significantly reduces retransmissions and delivery latency. However, since these protocols are only used to recover corrupted packets caused by lossy links, it is still unclear if such methods can be used to recover packets corrupted by collisions in WSNs.

In this chapter, we focus on how to efficiently recover collided packets in dense WSNs, with bursty and heavy collisions. First, we investigate PCP due to collisions through experiments on a testbed of Tmote sensors. Unfortunately, our experiment results show that almost all bits of collided packets are erroneous in typical WSNs where nodes use CSMA and send packets with similar sizes. These results indicate that current packet recovery methods cannot deal with the collision recovery. However, when nodes send packets with two different sizes, we observe that only certain bits in the long packet are corrupted in collisions between long and short packets, and this chunk of erroneous bits has almost the same size of the short packet. We call such collisions LS-collisions. LS-collisions produce a regular PCP in collided packets, which can be exploited to achieve efficient collision recovery. Therefore, such collisions are quite "useful". Our key insight is that if collisions cannot be totally eliminated under CSMA, it is better to have more LS-collisions, because we can recover collided packets from them.

In order to justify this idea, we provide a thorough analysis based on the model derived from scenarios where N senders compete for transmission using CSMA. The analysis demonstrates that by exploiting LS-collisions, we can achieve a higher *probability of successful transmission*, as well as a much higher *transmission efficiency* defined as the ratio of successfully transmitted information bits to overall bits transmitted. The analysis also gives insights on how to choose the right proportion of nodes sending long packets and short packets, and a sub-optimal probability distribution for senders to randomly select contention slots that maximizes the above two metrics.

We then present ACR, an *Active Collision Recovery* protocol that mitigates the negative impact of collisions by efficiently recovering collided packets. Unlike other existing recovery schemes, ACR is "active" in that it actively transforms potential collisions into LS-collisions to maximize the recovery probability, rather than passively waiting for collided packets to be recovered. ACR then uses a block based FEC scheme to efficiently recover corrupted packets due to LS-collisions. To identify erroneous blocks, ACR employs a novel RSSI-based error detection method, which does not introduce extra checksum overhead. In case of recovery failures, ACR also has a backup ARQ scheme. Unlike ZigZag [36] or PPR [46], ACR does not require customized hardware. ACR can be easily integrated into existing CSMA protocols with off-the-shelf sensor devices. Our main contributions are:

- An empirical study on bit error patterns in collided packets and the revelation of LS-collisions, which enables achieving efficient collision recovery.
- A theoretical analysis which demonstrates that we can achieve a significant performance improvement by exploiting LS-collisions under CSMA.
- A design of a novel ACR protocol, which actively creates LS-collisions by assembling packets from upper layers into long packets and short packets, and then uses a block-based lightweight FEC scheme to timely recover corrupted packets.
- A novel RSSI-based error detection mechanism to identify erroneous blocks with almost no overhead.
- Implementation and evaluation of the ACR prototype on a Tmote testbed. Results demonstrates that ACR achieves 25% higher transmission efficiency than existing recovery schemes.

. Our research results have been published in [105].

The rest of the chapter is organized as follows. Next, we present empirical results from experiments that investigate partial packet error patterns due to collisions in section 4.2. In section 4.3, we provides a theoretical study on the benefit of LS-collisions. In section 4.4, we present the details of our ACR design and implementation. In section 4.5, we evaluate the performance in a real testbed. Finally, in section 4.6, we conclude the paper.

4.2 Empirical Measurement

In this section, we present our measurement of bit error patterns in collided packets using CC2420 radios on Tmote-Sky motes, to answer the following questions: 1) Can we apply PPR schemes to recover collided packets? 3) In what type of collisions, will the collided bits be easy to recover?

4.2.1 Bit Error Pattern Comparison

We conduct a set of experiments on a testbed consisting of 48 Tmote-Sky motes to measure and compare bit error patterns in different cases. In the first set of experiments, we measure biterrors in corrupted packets due to poor link quality. We pick 5 pairs of senders and receivers, each of which is deployed in different rooms. The links between senders and receivers have poor qualities because of obstacles (walls), noise and multi-path fading. Every sender transmits 20 packets per second for 10 minutes, each with a payload of 100 bytes. Senders use different frequencies to avoid collisions. With the CRC check disabled, receivers can collect corrupted packets. We repeat the experiments 5 times to get average results. In the second set of experiments, we study the bit error pattern in collided packets. We select 6 close nodes, among which we pick one node as the receiver and the others as senders. Every sender transmits packets in the same way as the first experiments. This time all senders use the same frequency and start to send at the same time to generate collisions. Senders use the default CSMA protocol in TinyOS 1.x. We also repeat the experiments 5 times.

Figure 4.1 plots the cumulative distribution of the number of erroneous bits in corrupted packets. It is clear that bit error patterns are very different for the two scenarios. In the lossy link case, around 80% of the corrupted packets have less than 50 erroneous bits, which is only 6.25% of the whole packet. On the contrary, over 60% of the collided packets have more than 600 erroneous bits, which is 75% of all bits. This observation indicates that PPR methods in [46] [34] cannot achieve the same efficiency when recovering collided packets. First, since most bits are erroneous in collided packets,

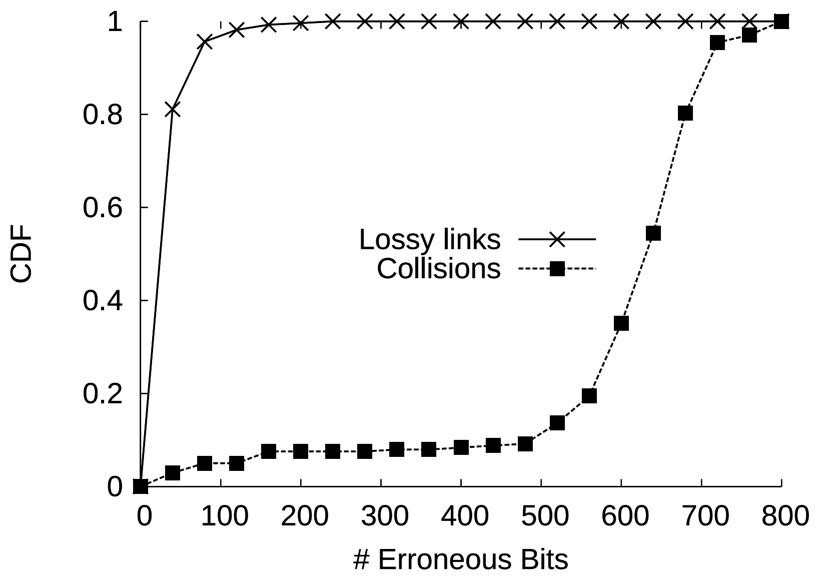


Figure 4.1: CDF of # bad bits in one packet caused by loss links and collisions

the performance gain of partial recovery is very limited. Second, since collisions cause more packet losses than corruptions, the chances to use partial recovery are limited. Actually, considering the cost and the complexity of the PPR implementation, it is more efficient to use the straightforward ARQ scheme for collision recovery.

Here, we also discuss why collisions generate such error patterns. In CSMA, nodes send packets only after sensing a free channel. Since all senders are close in our experiments, if one sender has been transmitting, others will wait. However, if two nodes sense a free channel at very close time points, they will send simultaneously and cause collisions. Such collisions produce two possible consequences. First, the Synchronization Headers (preamble and SFD) of both packets might be corrupted, which leads to packet loss at receivers. Second, even when a receiver synchronizes with one packet transmission, the signal of another packet may arrive shortly and start to corrupt the reception of the first one. Because two transmissions take almost the same time due to the same packet size, most bits of the first packet will be corrupted. CSMA already tries to eliminate such collisions, by making nodes randomly select different contention slots to sense the channel. However, when the number of transmission attempts grows, the likelihood of such collision increases. This collision can often be found in event-driven WSNs, or around sinks.

4.2.2 LS-Collision

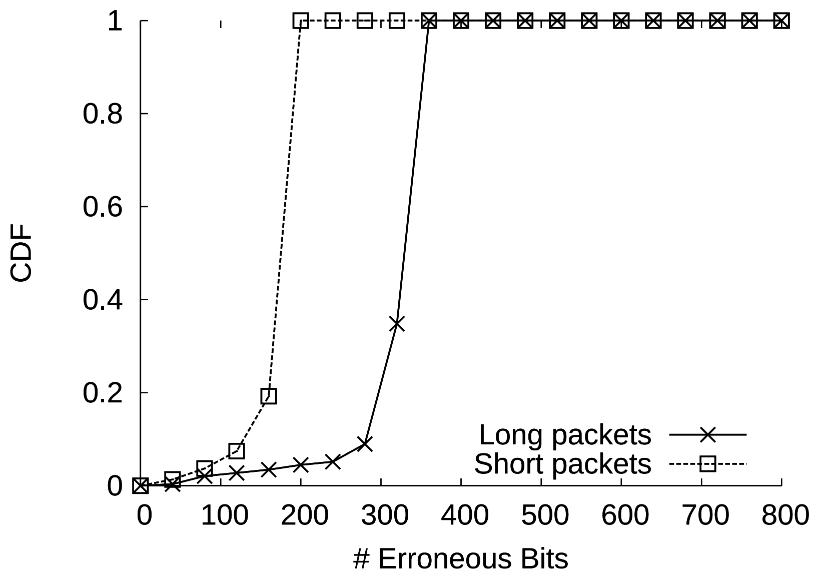


Figure 4.2: CDF of # bad bits in long and short packets

Our discussion shows that partial packet recovery cannot be used to recover packets corrupted by collisions. Are there some techniques that result in collisions that do meet these criteria? We have found such a technique. In the third set of experiments, we use one receiver and 5 senders, among which one sends packets with a payload of 100 bytes (typical for multimedia & medical readings in health care applications), and other senders transmit packets with a payload of 25 bytes (typical for climate readings in environment monitoring applications). All the other settings are the same as the second set of experiments. We also repeat this experiment 5 times. Each time, we use a different sender to transmit long packets. We measure the distribution of the number of erroneous bits in collided packets, separated by long packets and short packets, as shown in Figure 4.2. First, around 80% of the collided short packets have over 150 erroneous bits, 75% of the whole packet. This is a similar pattern with that of collided packets in the second set of experiments, where most bits in packets are corrupted. However, long collided packets show different patterns. In this experiment, over 90% of the collided packets have 300 to 360 erroneous bits, while most collided packets in the second experiment have more than 600 erroneous bits. Also, the number of erroneous bits in long collided packets is similar to the size of short packets (368 bits). We repeat the experiment with different long and short packet sizes and different mixtures of long and short packets, and observe the same phenomena. We call such collisions *LS-collisions*, which has two necessary conditions: 1) collisions occur between long and short packets; 2) receivers synchronize with long packets transmissions.

A LS-collision has some good features. First, when LS-collision occurs, a receiver still receives many correct bits in long collided packets. Second, the PCP in collided packets is predictable. The number of erroneous bits in collided packets is limited by the size of short packets, and most corrupted bits are continuous. Such features can help design efficient methods to recover long collided packets. For instance, we can design a FEC scheme to recover long packets. Since the number of erroneous bits is limited, it is easy to determine the appropriate degree of redundancy.

However, there are still challenging problems for exploiting LS-collisions. First, it should be justified whether exploiting LS-collisions can really improve the overall performance. Second, if LS-collisions are helpful, how can we have more LS-collisions instead of other collisions? How can we make nodes send different sizes of packets instead of the same size of packet as in most existing WSNs? How can we reduce collisions between long packets and between short packets? We will answer these questions in the following sections.

4.3 Model and Analysis

In this section, we first describe a simple CSMA model, and then provide an analysis to justify whether and to what extent LS-collisions can be exploited to improve the performance of such CSMA protocols, in terms of the probability of successful transmission and transmission efficiency. Further, we analyze the optimal carrier sense strategy that optimizes the performance of CSMA protocols combined with LS-collisions. Finally, we discuss the practical method to use LS-collision in real WSNs.

4.3.1 System model

We begin by describing the system model in typical WSNs, where nodes send packets on one channel using CSMA. We assume that at the beginning, there are N nodes attempting to send packets simultaneously and competing for transmissions. According to the CSMA implementation in TinyOS 1.x, the backoff time is divided into T contention slots. Each node picks a random contention slot $t \in [0, T]$. At contention slot t, the node carrier senses the medium. If the channel is free, it begins its transmission. Otherwise, it aborts or delays its transmission. If multiple nodes pick the same slot, they all sense a free medium and transmit at the same time, causing a collision. $P(t)$, $t = 0, 1, ..., T$, denotes the probability that a node picks the contention slot t. Obviously $0 \leq P(t) \leq 1$ and $\sum_{t=0}^{T} P(t) = 1$. We assume that each node independently selects the contention slot conforming to the same distribution. At last, D and H denote the length of the payload and the message header in packets, respectively. Next, we define two important performance metrics, the probability of successful transmission and the transmission efficiency.

Definition 4.1. *A node wins slot r if and only if it is the only one choosing slot r, and all others choose later slots. There is a* successful transmission *if and only if some node wins some slot in* $0, \ldots, T-1$. *Let* $\pi_p(N)$ *be the probability of successful transmission when N nodes select a contention slot using probability distribution p.*

Definition 4.2. *Let r be the ratio of useful bits correctly received divided by all bits sent by senders. The transmission efficiency* $E_p(N)$ *is the expected r when N nodes select a contention slot using probability distribution p.*

Two metrics we study here are crucial to the overall performance of a WSN. First, the probability of successful transmission can decide the channel throughput, and it can also impact the latency of packet delivery. A lower $\pi_p(N)$ indicates that nodes need to wait for more competition rounds to win the transmission chance. Second, the transmission efficiency is an indicator whether nodes use transmission energyefficiently. In energy-constrained WSNs, higher transmission efficiency is always desired.

Next, we consider a CSMA model with LS-collision recovery, called LS-CSMA model. First, we assume that there are two types of nodes, one type transmitting long packets and the other type transmitting short packets. We call them *long nodes* and *short nodes*, respectively. Let ρ be the fraction of long nodes among all nodes. Thus, the numbers of long and short nodes are $N\rho$ and $N(1-\rho)$. Let $P_L(t)$ and $P_S(t)$ be the probability distributions on slots for long and short nodes, respectively. Also, we use D_L,H_L and D_S,H_S to denote the length of the payload and the message header of long packets and short packets. We assume that when one long node and multiple short nodes pick the same slot i and all others choose later slots, a LS-collision occurs. Further, we assume that a FEC scheme is applied and F redundancy bits are added at the end of long packets, which guarantees that long collided packets can always be recovered after LS-collisions. Based on above assumptions, the probability of successful transmission should be modified.

Definition 4.3. *Under the LS-CSMA model, There is a* successful transmission *if and only if some node wins some slot, or a LS-collision occurs in some slot in* $0,\ldots,T-1$. *Let* $\pi_p^{LS}(N)$ *be the probability of successful transmission under LS-CSMA model. Also, let* $E_p^{LS}(N)$ *be the transmission efficiency under this model.*

4.3.2 Performance Gains from LS-collision

We now derive equations for $\pi_p(N)$ and $E_p(N)$ under pure CSMA model. First, the probability of some node wins in slot t is $NP(t)S(t+1)^{N-1}$, where $S(t+1)=\sum_{i=t+1}^{T}P(i)$ is the probability that the node selects any slot after slot t. Since the probability of successful transmission is the sum of all probabilities in each slot before slot T, we have We now derive equations for $\pi_p(N)$ and $E_p(N)$ under a pure CSMA model. First, the probability of some node wins slot t is $NP(t)S(t+1)^{N-1}$, where $S(t+1)=\sum_{i=t+1}^{T}P(i)$ is the probability that the node selects any slot after slot t. Since the probability of successful transmission is the sum of all probabilities in each slot before slot T, we have

$$\pi_p(N)=\sum_{t=0}^{T-1}NP(t)S(t+1)^{N-1} \tag{4.1}$$

This equation is also given in [95] [118]. For the transmission efficiency, since $r = D/(D+H)$ if successful transmission and $r = 0$ otherwise, we have

$$E_p(N) = \frac{D}{D+H}\sum_{t=0}^{T-1} NP(t)S(t+1)^{N-1} \tag{4.2}$$

Next, the following theorem studies these two metrics under the LS-CSMA model.

Theorem 4.1. *The probability of successful transmission and the transmission efficiency of LS-CSMA model are given by the following equations:*

$$\begin{aligned}\pi_p^{LS}(N) = \sum_{t=0}^{T-1}\Big[&N(1-\rho)P_S(t)S_S(t+1)^{N(1-\rho)-1}S_L(t+1)^{N\rho}\\ &+N\rho P_L(t)S_L(t+1)^{N\rho}S_S(t)^{N(1-\rho)}\Big],\end{aligned} \tag{4.3}$$

$$\begin{aligned}E_p^{LS}(N) = \sum_{t=0}^{T-1}\Big[&r_sN(1-\rho)P_S(t)S_S(t+1)^{N(1-\rho)-1}S_L(t+1)^{N\rho}\\ &+N\rho P_L(t)S_L(t+1)^{N\rho}\cdot\sum_{i=0}^{N(1-\rho)} r_iC^i_{N(1-\rho)}P_S(t)^iS_S(t)^{N(1-\rho)-i}\Big],\end{aligned} \tag{4.4}$$

where $S_S(t+1) = \sum_{i=t+1}^{T} P_S(i)$ *and* $S_L(t+1) = \sum_{i=t+1}^{T} P_L(i)$, $r_s = D_S/(D_S+H_S)$, $r_i = D_L/(D_L+H_L+F+i*(D_S+H_S))$, $0 \le i \le N(1-\rho)$.

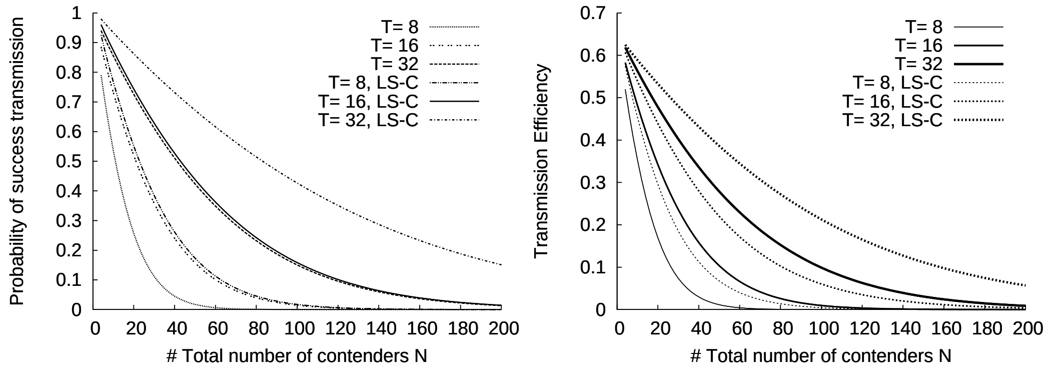
Proof. First, we prove Equation 4.3 under the LS-CSMA model. The successful transmission at slot t occurs in two cases. The first case is that one short node selects slot t and all other long nodes and short nodes choose later slots, whose probability is

$$N(1-\rho)P_S(t)S_S(t+1)^{N(1-\rho)-1}S_L(t+1)^{N\rho}.$$

When a long node selects slot t and all other long nodes select later slots, all short nodes can select slot t and we still get a successful transmission due to LS-collision recovery. The probability of this case is

$$N\rho P_L(t) S_L(t+1)^{N\rho} S_S(t)^{N(1-\rho)}.$$

Summing probabilities of all slot, we get Equation 4.3. For the transmission efficiency, r_i is the transmission efficiency ratio when one long node collided with i short nodes. It is easy to derive the expected transmission efficiency by computing the probability of every successful transmission case. ■



(a) Comparing Probabilities of Success Transmission (b) Comparing Transmission Efficiency

Figure 4.3: CSMA vs. LS-CSMA

These equations can be used to compare the performance of two models. Also, it can be seen that the probability on contention slots can greatly impact values of these two metrics. As the first step, we use the uniform distribution for slots selection, where $P(t) = P_L(t) = P_S(t) = 1/(T+1)$. The uniform distribution is easy to be implemented and is commonly used in most CSMA protocols.

We compare the performance of CSMA and LS-CSMA, with varying numbers of contenders. We assume that the numbers of long nodes and short nodes are the same, $\rho = 0.5$. Figure 4.3(a) shows successful transmission probabilities of two cases, computed by Equation 4.3. First, we can see all probabilities decrease to zero when N grows, and with T increasing, success probabilities increase. Second, it is very clear that LS-CSMA can always achieve higher success probabilities than CSMA under the same conditions. For example, when 20 nodes compete in 8 contention slots, LS-CSMA can improve the success probability from 25.2% to 55.1% over CSMA. From another perspective, if we set an acceptable threshold for the success probability, such as 70%, with 16 contention slots, CSMA can only contain around 10 nodes competing for transmission, but LS-CSMA can support 24 nodes and still meet the threshold. Third, LS-CSMA even slightly outperforms CSMA with double contention slots. This point can be found in two pairs, LS-CSMA with $T = 8$ and CSMA with $T = 16$, as well as LS-CSMA with $T = 16$ and CSMA with $T = 32$. This surprising observation indicates that LS-collision recovery can help CSMA achieve the same, even higher success probability by using half of the number of slots. Above all, LS-CSMA can provide more reliable transmission, alleviate transmission contention, and reduce packet delivery latency by using much less contention slots.

We also look into the transmission efficiency. We assume that under the CSMA model, nodes send packets with a payload of 25 byte and a header of 10 bytes. For the LS-CSMA model, short nodes send the same packets following the CSMA model. Long nodes send packets with a payload of 80 bytes, a header of 10 bytes, plus 35 bytes FEC redundant bits. Here, we assume these redundant bits are always sufficient to recover collided packets. We can see that under the same conditions, LS-CSMA always achieves higher transmission efficiency. For example, with 20 nodes competing in 16 slots, LS-CSMA improves efficiency from 0.34 to 0.43 over CSMA. Note that the transmission ratio for a short packet is around 0.71, but that is 0.64 for a long packet, because long packets carry redundant bits for potential recovery. So, if every packet is transmitted correctly without collisions, CSMA should achieve higher efficiency than LS-CSMA. But because LS-CSMA can recover long packets from LS-collisions, and be more likely to get successful transmissions, it achieves higher efficiency in the end. Furthermore, if we also include the cost of retransmissions into computing transmission efficiency, the gap between these two would be enlarged, because CSMA has a lower success probability, causes more retransmissions and retransmission is expensive in term of extra bits transmitted.

Above results demonstrate the benefits of combining LS-collisions into CSMA protocols. With the commonly used uniform distribution, LS-CSMA outperforms CSMA in both metrics. However, as shown in [95] [118], the uniform distribution is not the optimal slot selection strategy for CSMA. The probability of successful transmission can be further improved by using other distributions.

4.3.3 Slot Selection Distribution

In this subsection, we focus on how to optimize the probability of successful transmission $\pi_p^{LS}(N)$, according to Equation 4.3. We assume that all variables are given, except the distributions $P_L(t)$ and $P_S(t)$. The goal is to find the optimal distribution for both long nodes and short nodes.

We propose a two-step method to approach to the optimal. This two-step method comes from one observation. In order to achieve higher successful transmission, the worst situation we want to avoid is that multiple long packets make collision at slot t. In that case, no matter what short nodes select, there is no successful transmission. Since long nodes and short nodes pick slots independently, our first step is to find the optimal $P_L(t)$ which minimizes the probability that multiple long nodes collide at some slot. In the first step, we do not consider short nodes' behaviors. Let P_L^* denote this optimal distribution. At the second step, given the probability of long nodes, we can find the optimal distribution of short nodes, denoted by P_S^*. . The optimal distribution P_L^* can be given by a recursive way. Let $S_L^*(t) = \sum_{i=t}^{T} P_L^*(i)$, and $K_L^*(t) = \frac{S_L^*(t)}{S_L^*(t-1)}$, where $1 \leq t \leq T$. We can get the recursive formulas for $S_L^*(t)$ and $K_L^*(t)$ as follows.

$$S_L^*(t+1) = K_L^*(t+1) S_L^*(t),$$

where $t = 0, ..., T-1$, and $S_L^*(0) = 1$.

$$K_L^*(T-t-1) = \frac{N_l - 1}{N_l - K_L^*(T-t)^{N_l - 1}},$$

where $t = 0, ..., T-2$, $N_l = N\rho$ is the number of long nodes and $K_L^*(T) = \frac{N_l - 1}{N_l}$.

Therefore, the optimal distribution $P_L^*(t)$ is

$$P_L^*(t) = S_L^*(t) - S_L^*(t+1),$$

where $t = 0, ..., T-1$, and $P_L^*(T) = S_L^*(T)$.

Now given the probability of long nodes, we compute the optimal distribution of short nodes, also defined as the recursive formulas.

		$P^*(0)$	$P^*(1)$	$P^*(2)$	$P^*(3)$	$P^*(4)$	$P^*(5)$	$P^*(6)$	$P^*(7)$	$P^*(8)$	
$N = 16$	Long node	0.0265	0.0290	0.0321	0.0361	0.0415	0.0495	0.0627	0.0903	0.6323	$\pi_p^{LS}(16) = 0.89$
	Short node	0.0175	0.0195	0.0220	0.02553	0.0305	0.0384	0.0535	0.0991	0.6938	

		$P^*(0)$	$P^*(1)$	$P^*(2)$	$P^*(3)$	$\cdots$	$P^*(29)$	$P^*(30)$	$P^*(31)$	$P^*(32)$	
$N = 64$	Long node	0.0018	0.0019	0.0020	0.0020	$\cdots$	0.0130	0.0173	0.0268	0.8300	$\pi_p^{LS}(64) = 0.967$
	Short node	0.00120	0.00120	0.0012	0.0013	$\cdots$	0.0099	0.0143	0.0281	0.8724	

Table 4.1: Examples of the Optimal Distribution. (TOP) Optimal Distribution for $T = 8$. (BOTTOM) $T = 32$

Theorem 4.2. *Given the distribution of long nodes, P_L^*, the optimal distribution of short nodes that maximize the probability of successful transmission under LS-CSMA model, P_S^* can be derived as follows.*

$$K_S^*(T-t-1) = \frac{N_s - 1}{N_s - K_L^*(T-t)^{N_l-1}[N_l(1-K_L^*(T-t)) + K_L^*(T-t)K_S^*(T-t)^{N_s-1}]} \tag{4.5}$$

$$\pi_p^{LS}(N) = (1-P_S^*(0))^{N_s-1} \cdot (1-P_L^*(0))^{N_l-1} \cdot [N_l - (1-P_L^*(0)) * (N_l - 1)] \tag{4.6}$$

$$K_S^*(T) = \frac{(N_s-1)[r_s K_L^*(T) + r_1 N_l(1-K_L^*(T))]}{r_s N_s K_L^*(T) + (r_1 N_s - r_0) N_l (1-K_L^*(T))} \tag{4.7}$$

Let $S_S^*(t) = \sum_{i=t}^{T} P_S^*(i)$, and $S_S^*(t+1) = K_S^*(t+1)S_S^*(t)$, where $t = 0, ..., T-1$, and $S_S^*(0) = 1$. Then $K_S^*(T-t-1)$ is given in Equation 4.5 and

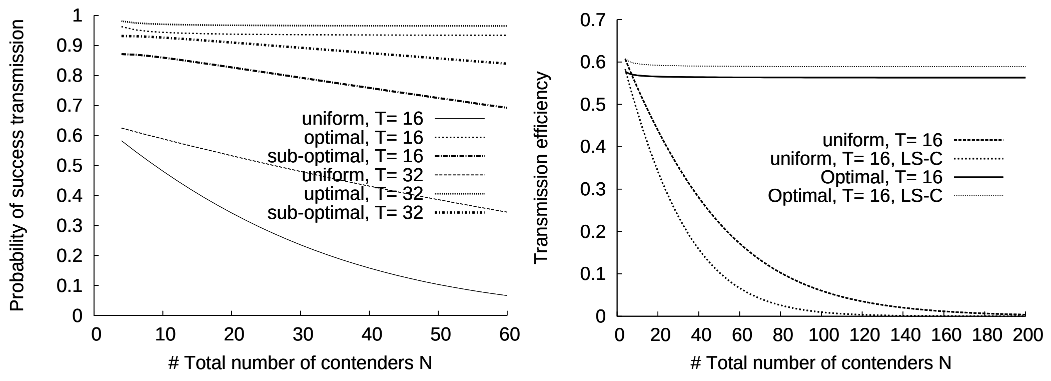
$$K_S^*(T) = \frac{N_s - 1}{N_s}$$

where $N_s = N(1-\rho)$ is the number of short nodes, $t = 0, ..., T-2$. Therefore, the optimal distribution $P_S^*(t)$ is

$$P_S^*(t) = S_S^*(t) - S_S^*(t+1),$$

where $t = 0, ..., T-1$, and $P_S^*(T) = S_S^*(T)$. With this distribution, the maximum probability of successful transmission, $\pi_p^{LS}(N)$ is given by Equation 4.6

With the above theorem, we can compute the optimal distributions for long nodes and short nodes. Table 4.1 shows some examples of the optimal distribution. First, we can find that optimal distribution of long nodes and short nodes are very similar. Both probabilities increase slowly at the first few slots, but increases rapidly for last few slots, especially when the number of nodes is large. We can see that for large N, the most probability is concentrated on the last slot, which can reduce the number of nodes that pick previous slots and thus improve the probability of a successful transmission. Comparing $P_S^*(t)$ with $P_L^*(t)$, we find that in most cases, the probability of short packets is slightly less than that of long packets at all first $T-1$ slots, and the probability of short packets is more concentrated at the last slot. Its effect is to avoid the situation that multiple short packets pick slot t and all long nodes pick later slots. Finally, we can see that when the number of contention slots is relative large (32), we can achieve a very high probability of successful transmission, even when the number of nodes is very large. For example, when $T = 32$, even with 192 nodes, the probability of successful transmission can still reach 0.96.



(a) Probabilities of Success Transmission (b) Transmission Efficiency

Figure 4.4: Optimal distribution vs. Uniform distribution

Figure 4.4(a) compares the performance of the optimal distributions and the uniform distributions under two models. We can see that the optimal distributions on both models achieve much higher probability of successful transmission than the uniform distribution, especially when the number of contenders becomes large. Another nice feature of the optimal probability is that it can keep high probabilities of successful transmission, no matter how many nodes is competing. Third, comparing the optimal distributions of two models, LS-CSMA can still achieve a higher transmission-success probability than CSMA, both with the optimal distributions. However, the performance gap between two models is diminished, since both achieve high success probabilities with the optimal distributions.

Figure 4.4(b) shows the transmission efficiency computed with this distribution. We use the same setting in Figure 4.3(b). We can see this figure exhibits the similar trend with Figure 4.4(a), where this distribution work much better than the uniform distribution, and keep a high transmission efficiency, even when the number of contenders is large. Also, with the optimal distribution, LS-CSMA improves the transmission efficiency from 0.52 to 0.59 over CSMA. Note that this distribution does not yield the optimal transmission efficiency in Equation 4.4. We leave finding the optimal distribution for transmission efficiency as a future work.

So far, we find a better carrier sense strategy for both long and short nodes. The numerical analysis demonstrates that this distribution improves the performance of LS-CSMA in both metrics. Another good feature is its good scalability. Results show that this distribution maintain near-constant performance, when the number of contenders grows rapidly. However, computing the optimal distribution requires the knowledge of the number of long nodes and short nodes who are competing for transmission. These numbers may vary from time to time, especially in event-driven WSNs. Also, the process of computing the distribution is very complicated and too costly for power-limited sensor nodes. Therefore, we want to find alternative distributions which provide near-optimal results and can be computed by simple solutions without the need of exact information about other contenders.

4.3.4 Practical Approach

In this subsection, we discuss how to design a practical solution to find a good distribution for nodes to select contention slots. From the observation of the optimal probability distribution, we find two important properties. First, the last few slots take a lot of probability, while the probabilities on the first few slots are very small. Second, short nodes share similar distribution with long packets, but short nodes have slightly higher probabilities to select the last few slots than long nodes. The first observation suggests using an increasing geometric sequence to compute a sub-optimal distribution.

Works in [118] also use the geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{4.8}$$

where $t = 0, ..., T$, and b is a number greater than 1. In this geometric sequence, the increasing ratio is $b^{\frac{t}{T+1}}$. One key decision is which value of b should be chosen. Authors in [118] argue that a larger value should be chosen. For example, authors show that when there are 33 contention slots, one can choose $b = 1000$, and get the no-collision probability smaller than the optimal result by only 6%. The underlying reason is that by using a large b, nodes tend to select the last few slots, thus reduce the chances of make early collisions. This approach works very well under CSMA, especially with a large number of contenders. However, we find that a large value of b is not a good choice for our LS-CSMA model.

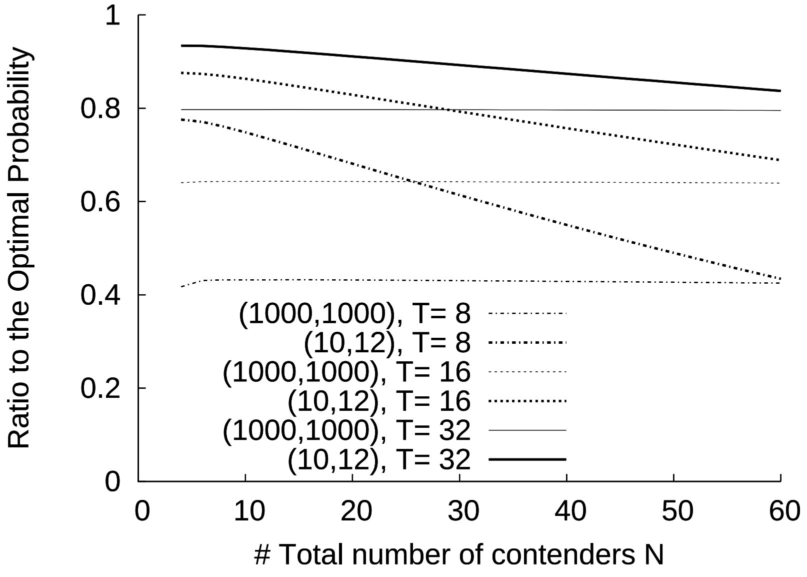


Figure 4.5: Two Geometric Sequences

In Figure 4.5, we compare two geometric distribution in terms of their probabilities relative to the optimal result. In one distribution, we use $b = 1000$ to compute the sequence, according to Equation 4.8, and make short nodes and long nodes use the same distribution. Second, we use a small value of b, where $b = 10$ for long nodes, and $b = 12$ for short nodes. The result shows that the geometric sequence with smaller b values is better than that with large b values. The reason is that a large b reduces the chances of nodes colliding at early slots, but also diminish the likelihood of LS-collision. When the number of contender is not very large, the reduction of early collisions is not obvious, and the loss of a large b overcomes the performance gain. The second sequence with a small b are very close to the optimal result with 32 contention slots. For example, with 20 nodes, its ratio to the optimal result is 0.93. When contention slots are less than 32, this sequence can still work well, when the number of contenders is not too high. Consider that the number of senders in one neighborhood is usually less than 30, such a sequence offers us near-optimal performance in most cases. Furthermore, this geometric sequence can be simply computed with the following equation:

$$i = \lfloor (T+1)\log_b[\alpha(b-1)+1] \rfloor \tag{4.9}$$

where α is a random variable with a uniform distribution within the interval $[0,1)$, and $b = 10$ for long nodes or $b = 12$ for short nodes.

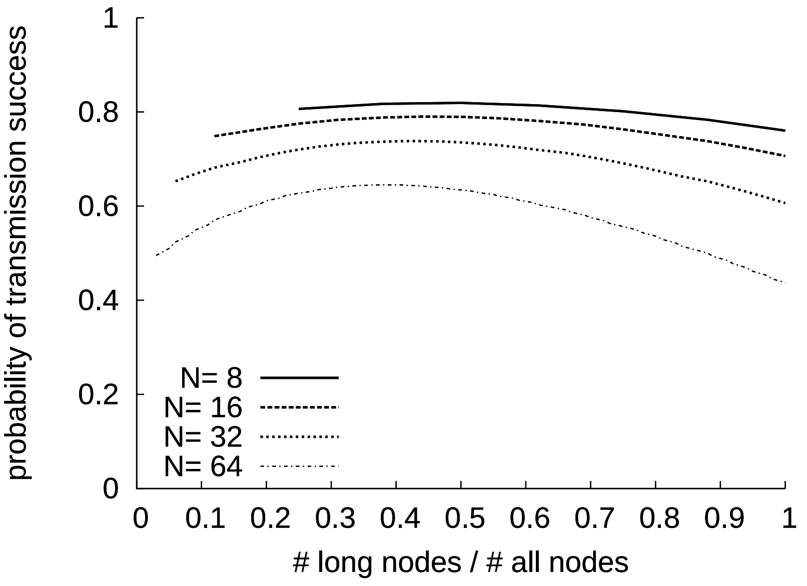


Figure 4.6: Proportion of long and short nodes

The last problem is how to determine whether a node should be a long or short node. The basic idea works as follows. When a node has messages to send, it will “toss” the coin to make the decision, according to a pre-defined probability. In other words, we need to determine the right value of ρ. We compute the probability of successful transmissions, with different values of ρ. Results are shown in Figure 4.6. We can see that the success probability changes smoothly while ρ changes. All maximum values appear in the middle of the curve, which means the best value of ρ is around 0.5. That makes sense, because if unbalanced, nodes of the majority are likely to generate more collisions among themselves. Such long-long, or short-short collisions will cause transmission failures and are not preferred in the LS-CSMA model.

4.4 ACR design

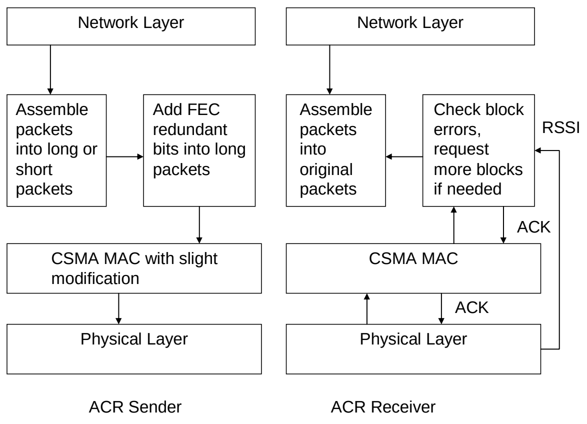

Network Layer
Network Layer
Assemble packets into long or short packets
Add FEC redundant bits into long packets
Assemble packets into original packets
Check block errors, request more blocks if needed
RSSI
ACK
CSMA MAC with slight modification
CSMA MAC
ACK
Physical Layer
Physical Layer
ACR Sender
ACR Receiver

Figure 4.7: ACR Components

In this section, we present the design of ACR, the *Active Collision Recovery* protocol, which exploits the advantages of LS-collisions to efficiently recover collided packets. ACR is implemented as a link layer protocol combined with a slight modification on CSMA protocols. Figure 4.7 illustrates the functions of ACR at both the sender and the receiver sides.

4.4.1 ACR Sender

A sender node of ACR actively converts potential collisions into LS-collisions and then applies hybrid schemes to recover corrupted packets from collisions. At senders, ACR protocol has the following steps:

4.4.1.1 Packet Assembly

One key condition for LS-collision is the co-existence of long and short packets. Thus, Packet Assembly (PA) step actively assembles packets from the network layer into long packets or short packets, and feeds them into the MAC layer. PA needs to determine if nodes should send long or short packets, by "tossing a coin" with the probability ρ. Here, we use 50%, according to the analysis in Section 4.3.4. Another important problem is to select the right sizes for long and short packets, which depends on different radio hardware and collision conditions. In this paper, we focus on off-the-shelf mote platforms, such as MicaZ or Tmote-Sky, which can send a packet as long as 128 bits. We select packet sizes of 25 and 80 bits for short and long packets.

4.4.1.2 FEC and CRC

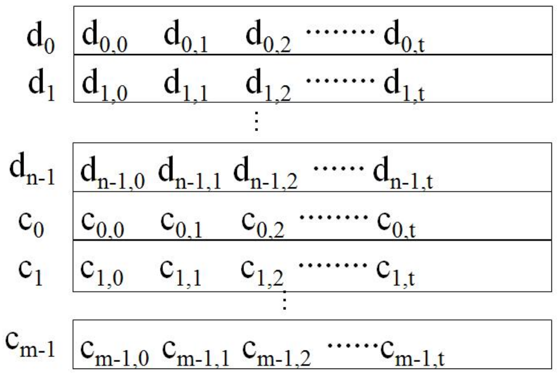

d_0 $d_{0,0}$ $d_{0,1}$ $d_{0,2}$ ········ $d_{0,t}$
d_1 $d_{1,0}$ $d_{1,1}$ $d_{1,2}$ ········ $d_{1,t}$
d_{n-1} $d_{n-1,0}$ $d_{n-1,1}$ $d_{n-1,2}$ ······ $d_{n-1,t}$
c_0 $c_{0,0}$ $c_{0,1}$ $c_{0,2}$ ········ $c_{0,t}$
c_1 $c_{1,0}$ $c_{1,1}$ $c_{1,2}$ ········ $c_{1,t}$
c_{m-1} $c_{m-1,0}$ $c_{m-1,1}$ $c_{m-1,2}$ ······ $c_{m-1,t}$

Figure 4.8: Illustration of the FEC scheme

If nodes are to send short packets, ACR adds a CRC code at the end of each short packet. Later, receivers use this CRC as a checksum to verify the integrity of the packet. In implementation, we applies a 16-bits CRC. If nodes are to send long packets, ACR uses FEC codes to allow receivers to recover long packets without retransmission when LS-collisions occur, which reduces communication overhead as well as recovery latency. The basic idea is to divide a long packet into n blocks, and then compute m redundant blocks attached at the end of packets, so that a receiver can still reconstruct the original packets until more than m blocks are corrupted. One traditional way is to compute $n+m$ linear combinations of the original n blocks, and assemble these $n+m$ linear combinations to a packet to transmit. Any correct n combinations are sufficient for receivers to reconstruct the original data. However, such a method requires a large amount of matrix operations, not suitable to Tmote nodes. Instead, we take advantage of the special error patterns of LS-collisions to design a lightweight FEC scheme. Figure 4.8 shows an example of how to encode a packet. Here, a packet is divided into n blocks, $d_0 \cdots d_{n-1}$. m redundant blocks are computed by

$$C_{0,j} = d_{0,j} \oplus d_{1,j} \oplus \cdots \oplus d_{n-1,j},$$

$$C_{k,j} = d_{k,j} \oplus d_{k+m,j} \oplus \cdots \oplus d_{k+m \cdot \lfloor \frac{n-1}{m} \rfloor, j},$$

where $0 \leq j \leq t$, $1 \leq k \leq m-1$ and $\oplus$ denotes Exclusive OR operations. As observed in Section 4.2, the number of corrupted bits in collided packets is not greater than the size of short packets, denoted by S. Thus, we set the size of redundant bits $(t+1)m >= S$, which guarantees that redundant bits are always more than corrupted bits. For the second observation, corrupted bits are continuous in collided packets, therefore those bits corrupt at most m continuous blocks, $d_j \cdots d_{j+m-1}$. Our encoding method guarantees that any two blocks within these m continuous blocks are not both used to generate one redundant block, except C_0. Therefore, this FEC method can always recover collided packets due to LS-collisions. This FEC method only requires bit operations for encoding and decoding, and requires no extra storage. Thus it can be implemented easily in sensor nodes.

Another problem is to determine the right size of blocks. We apply various sizes of blocks to collided packets in a real trace file, and measure the number of redundant bits (block_size × No. redundant block) necessary for recovery, as shown in Figure 4.9. Generally, the smaller blocks are, the less redundancy we need. The reason is that larger blocks are likely to cause more "wasted" bits, which are correct, but in corrupted blocks. As a result, smaller blocks are preferred. The size of blocks also depends on the method to identify erroneous blocks. As in [34], one extra byte of checksum is attached to each block for error detection. With smaller sizes of blocks, more blocks introduce more overhead. Therefore, the block size is bounded within 20∼25 bytes in [34]. However, ACR applies a novel method for error detection, which efficiently identifies erroneous blocks without extra checksums, thus supports even smaller blocks. With this method, we use a

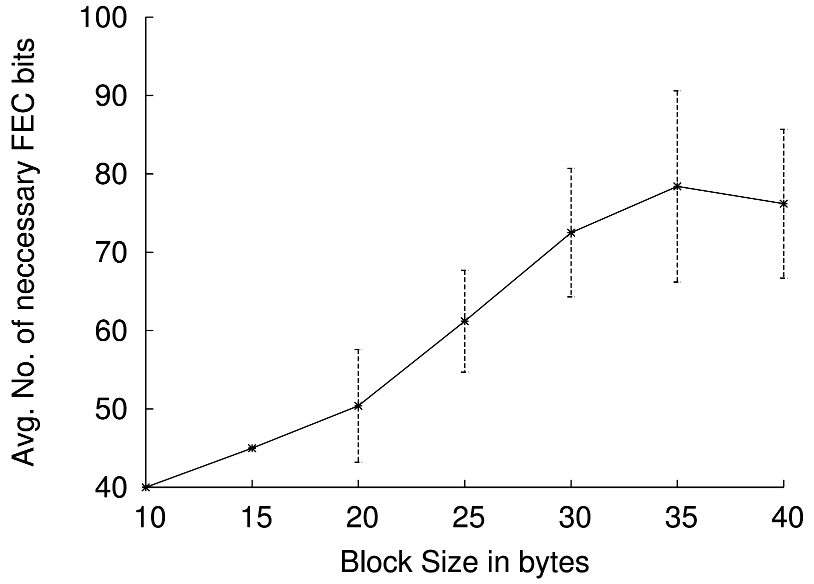


Figure 4.9: Using Small blocks is more efficient

block as small as 10 bytes. We will introduce this method later.

After adding the FEC codes, we also add 16 bits of CRC code at the end of packets, which are computed based on the original packets without FEC bits.

4.4.1.3 Modified CSMA

ACR slightly modifies a CSMA protocol to further increase the chances of LS-collisions. There are two changes.

Non-uniform Distribution on contention slots. Even with long packets and short packets, contenders can still end up with collisions other than LS-collisions, such as collisions between long packets. According to Section 4.3.4, we can use Equation 4.9 to compute a non-uniform distribution for senders, which increases the likelihood of LS-collisions and results in better performance. This only requires a minor change on the underlying MAC layer (such B-MAC, S-MAC), replacing the uniform backoff distribution of those protocols.

Transmission Delay on short packets. Another necessary condition for LS-collision is that receivers should get synchronized with long packets' transmission first. In order to realize this condition, we introduce a slightly delay on short packets' transmissions. When sensing a channel clear, short packets wait for a time τ, while long packets do CCA immediately. Here, τ is equal to the time to transmit the synchronization header of a packet, which is roughly 130 μs in CC2420 radios. This time is shorter than one contention slot (e.g. 320μs in TinyOS), thus it does not impact CSMA protocols.

4.4.1.4 Backup ARQ scheme

At last, ACR also provides a backup ARQ scheme to recover lost packets in three cases. 1) Packets lost due to other kinds of collisions; 2) short packets lost in LS-collisions; 3) long packets that have no sufficient correct blocks to be recovered by the FEC scheme. In this ARQ scheme, a sender waits for ACK after a transmission. If ACK is not received after a timeout, it retransmits the previous packet. If ACK is received with notification that several blocks are needed for recovery, it retransmits that number of blocks, instead of the whole packet. This ACK scheme brings extra overhead by retransmission. But since the FEC scheme can recover most long collided packets from LS-collision, the number of retransmissions is less than that of the pure ARQ.

4.4.2 ACR Receiver

When receiving packets, ACR first determines if they are short or long packets. If it is short, ACR uses the attached CRC to determine if the packet is correct or not. If the CRC check is passed, the receiver sends an ACK message.

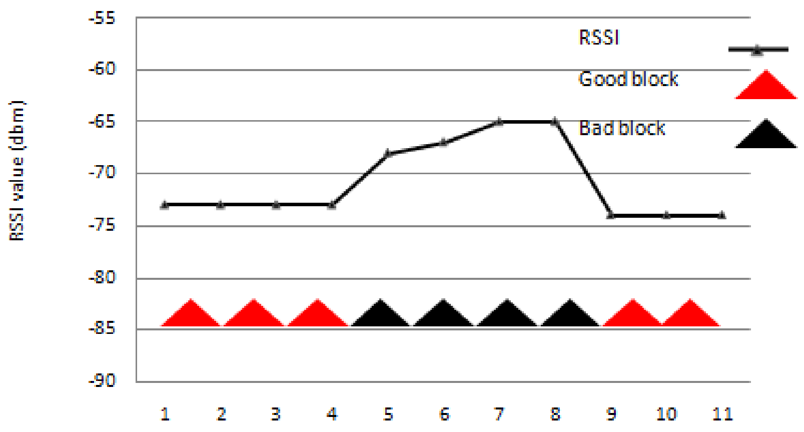


Figure 4.10: Relation between RSSI and bad blocks

If nodes receive long packets, ACR also uses the CRC to determine whether there are any erroneous blocks in this long packet. If the packet passes the CRC check, receivers send a success ACK. Otherwise, receivers know some blocks are corrupted. Next, ACR needs to identify erroneous blocks. As mentioned before, ACR does not add extra bits on each block to detect corruption, because it costs extra overhead. Instead, we introduce a new method. This new method is based on the fact that during reception, RSSI values at the receiver side increase when collisions occur, but be almost constant without collision. We conduct experiments to measure the RSSI value in the course of packet reception. One typical result is shown in Figure 4.10. We can clearly see that the RSSI value increases to above $-68dBm$ when collisions corrupt blocks. Without collisions, the RSSI value is stable at $-73dBm$. Thus, we can detect a corrupted block by checking whether the RSSI value during the reception of this block is high or normal. In our method, a receiver continuously samples RSSI values every δ time units and keeps them in an array, as soon as a reception begins. Later after the reception is over, ACR looks up the array to seek the instant increase of RSSI values. If it finds any high value, it maps the value to the corresponding block, and marks it as a potential corrupted block. Here time δ should be equal to the time of receiving one block. We implement this RSSI based error detection method in Tmote nodes, and find that the maximum RSSI sampling rate that hardware can support is one reading per $200 \sim 250 \mu s$, which means that this method can support the block as small as 8 bytes without extra overhead. Considering other factors, we choose the block size as 10 bytes. By using this method, we can identify most corrupted blocks in our experiments. Occasionally, this method missed some bad blocks at the beginning of the collision. Then ACR cannot recover the long packet and it uses the backup ARQ to recover them.

After erroneous blocks are found, ACR firstly checks if the number of erroneous blocks is greater than m. If so, the FEC scheme cannot recover the blocks, and ACR sends an ACK message with a block bitmap to request senders to send good blocks. Note that it is not necessary to send all erroneous blocks, instead ACR only needs good blocks until the number of erroneous blocks is not greater than m. With less than $m+1$ erroneous blocks, ACR use the FEC codes to recover these blocks. For a corrupted block d_k, suppose $k = i \cdot m + r$, if $r \neq 0$, then this block are reconstructed by

$$d_{k,j} = c_{r,j} \oplus d_{r,j} \cdot\cdot \oplus d_{(i-1)m+r,j} \oplus d_{(i+1)m+r,j} \cdot\cdot \oplus d_{\lfloor \frac{n-1}{m} \rfloor m+r,j},$$

If $r = 0$,

$$d_{k,j} = c_{0,j} \oplus d_{0,j} \cdot\cdot \oplus d_{k-1,j} \oplus d_{k+1,j} \cdot\cdot \oplus d_{m-1,j},$$

where $0 \leq j \leq t$. Here, we should always compute blocks with $r \neq 0$ first, and them reconstruct the block with $r = 0$. After recovering corrupted blocks, ACR conducts the CRC check, and sends an ACK if it passes the CRC check.

4.4.3 Discussion

ACR is designed to deal with collisions when multiple nodes are sending packets simultaneously. As mentioned before, this scenario occurs frequently in dense WSNs, when events trigger burst-traffic, or when high-volume traffic is aggregated at sinks. However, collisions may occur very rarely when events do not appear in WSNs. In that case, CSMA protocols like B-MAC, are sufficient to avoid collisions, and using ACR brings extra overhead. Therefore, it is better to combine CSMA and ACR into an adaptive protocol, which applies CSMA when there are few collisions, but switches to ACR when multiple nodes start to compete for transmissions. This adaptive protocol can be a promising protocol to achieve efficiency under various scenarios.

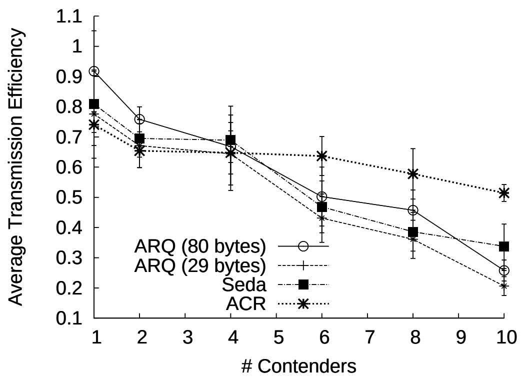
The second concern is the efficiency of packet assembling. People may argue that assembling packets may bring overhead by segmenting large packets into small ones. But in current WSNs, most packets tend to be small, so most assembling actions are to aggregate small packets into large ones. Furthermore, our analysis in Section 4.3 shows that ACR achieves better transmission efficiency than CSMA.

Another design concern is energy efficiency. Energy Consumption in WSNs primarily comes from idle listening. ACR does not directly focus on how to reduce the idle listening, but ACR is compatible with MAC protocols that address this issue, such as B-MAC [80], mainly because ACR does not demand any nodes to overhear or receive extra signals except receivers.

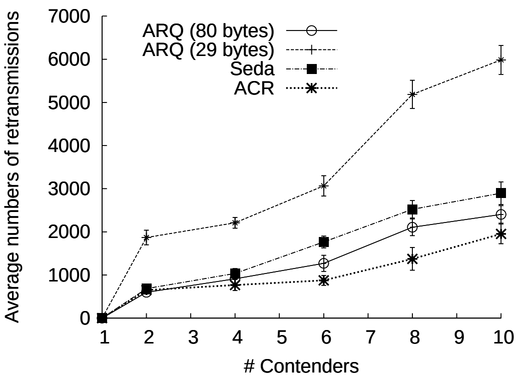
4.5 Performance Evaluation

In this section, we evaluate the performance of ACR scheme on our testbed to establish the effectiveness of our scheme in reality. The main scenario we consider is that multiple nodes have a certain amount of information to transmit to one sink. In our test bed, all nodes can send packets to the sink in one hop. We compare ACR with two other types of packet recovery schemes, pure ARQ and Seda [34]. In Seda, a long packet is divided into blocks. Receivers send ACKs to request senders to retransmit corrupted blocks. Unlike ACR, Seda adds one byte CRC and one byte sequence number to each block for error detection. Default values used in our experiments are as follows. Every sender transmits at a power of $-10dBm$, which ensures good links. The senders have 400 bits of information to send every second. We consider two packet sizes for pure ARQ, namely 29 and 80 bytes. For Seda, the data size of one packet is 80 bytes, divided into 4 blocks, each of which have 20 bytes of data, as suggested in [34]. For ACR, short packets have 25 bytes of data, and long packets have 80 bytes data, divided into 8 blocks, with a block size of 10 bytes. It also carries 4 blocks of redundancy bits. For all three methods, we shorten the MAC header, and make the length of the whole header 6 bytes. Also, the ARQ times out after $2ms$. At last, all methods use the default CSMA protocol in TinyOS 1.x, with 16 contention slots. All experiments run for 10 minutes and send 4000 bits. Each data point is obtained by averaging results of 5 trials. We also add 90% confidence intervals for each data point.

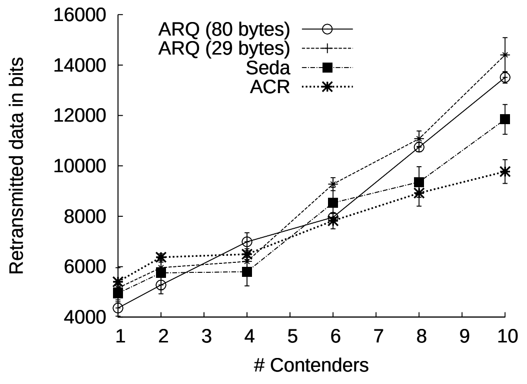
Figure 4.11(a) plots the measured transmission efficiency of all methods with various numbers of senders, calculated by the information bits transmitted over all bits transmitted or retransmitted. We observe that ACR has the lowest efficiency when there is only one contender with no collisions, because the long packets of ACR bring extra redundant bits. However, when the number of contenders grows, ACR starts to recover long collided packets from LS-collisions and outperforms other schemes. This result implies that ACR is more efficient to recover collisions than other methods. For example, with 6 contenders, ACR improves by $\sim 25\%$ the transmission efficiency over ARQ. On the contrary, Seda has the lowest efficiency when collisions increase. This is mainly because that collided packets do not have a partial packet pattern, so Seda needs retransmit the whole



(a) Transmission Efficiency



(b) # Retransmissions



(c) Volume of data retransmitted

Figure 4.11: Recovery performance under various # contenders

packet. Further, Seda pays extra overhead of checksums.

Figure 4.11(b) plots the number of retransmissions with various number of senders. We can see that with heavy collisions, ACR triggers much fewer retransmissions than others. For example, with 8 contenders, ACR reduces retransmissions by 42% than ARQ with long packets. This reduction comes from two places. First ACR applies a better distribution to select contention slots, which leads to fewer collisions. Second, ACR generates LS-collisions, and uses a FEC method to recover long collided packets from such collisions, thus it only needs to retransmit short collided packets. Other methods have to retransmit all collided packets after collisions.

Figure 4.11(c) plots the number of bits retransmitted with various number of senders. This figure demonstrates that with heavy collisions, ACR retransmits much fewer bits than others. For example, with 10 contenders, ACR reduces retransmitted bits by 17.4% than Seda. As discussed before, the main reason of this reduction is that ACR recovers most long packets without retransmission, therefore it saves a large number retransmitted bits.

4.6 Conclusion

This paper explores solutions to efficiently recover collisions in WSNs. Through empirical experiments, we first discover that LS-collisions have nice partial corrupted patterns in collided packets. Then, we propose ACR that actively converts potential collisions into LS-collisions and also uses lightweight FEC to recover collided packets from LS-collisions. A novel RSSI-based error detection method is also developed in ACR. We present theoretical analysis that shows that ACR greatly improves the probability of successful transmissions and enhances the transmission efficiency. We evaluate ACR performance in TinyOS with Tmote-Sky motes, and we demonstrate that ACR is more efficient in recovering collisions than existing solutions, by significantly reducing the number of retransmissions and increasing the transmission efficiency.

Chapter 5

Run Time Assurance of Application-level Reliability

Emerging wireless sensor network (WSN) technologies are applicable to a wide range of mission-critical applications, including fire fighting and emergency response, infrastructure monitoring, military surveillance, and medical applications. These applications must operate reliably and continuously due to the high cost of system failure. However, continuous and reliable operation of WSNs is notoriously difficult to guarantee due to hardware degradation and environmental changes, which can cause operating conditions that were impossible for the original system designers to foresee. This is particularly true for applications that operate over long time durations, such as a building monitoring system that must operate for the lifetime of the building, which may be more than 100 years. Wireless noise and interference may change dramatically as new wireless technologies are developed and deployed in or near a building, and sensor readings and network topology may change as the occupancy, activities, and equipment in a building evolve over time. Furthermore, a WSN system usually consists of many components. Even each component employs fault-tolerant or other mechanism to achieve high reliability. The system may still fail because of wrong or unexpected synergistic behaviors. To address this dilemma, there must be some methodologies that directly improve and maintain the application-level reliability of WSN systems.

We propose and demonstrate a methodology for *run-time assurance* (RTA). We use program analysis and compiler techniques to automate WSN code generations according to application-level requirements and facilitate automated testing of a WSN at run time. As a proof of concept, we implemented a framework for designing and automatically testing WSN applications. We evaluate our implementation on both simulation and real testbed, and compare performance with an existing network health monitoring solution. We have conducted extensive research on this topic [105]. Our work answer the following research questions in this chapter:

- How can Run-time Assurance improve and maintain the application-level reliability? What are the basic principles and methodologies?
- How to specify the application level semantics and RTA requirements?
- How to design and implement WSN codes that satisfy application level and RTA requirements? How to make this code generation automated?
- How to automatically run tests and check results? How to make the impact of tests minimal? Specifically, how to execute tests with less overhead and resource occupancy, as well as to correctly deal with collisions between real events and tests?
- How to collect useful information to pinpoint the failures if RTA tests fail?

This chapter is organized as follows: we present the RTA concept and design principles in Section 5.1. Section 5.2 describes in more detail the main components of our RTA framework - the SNEDL programming model, the automated test and code generation, and the test execution support. Next, a case study is presented in Section 5.3 and the evaluation results are shown in Section 5.4. Finally, Section 5.5 concludes the paper.

5.1 RTA Principle

Significant effort has been applied to build reliable WSN systems. Advanced and rigorous testing and debugging techniques are used to verify WSN systems before deployment. In addition, many systems today employ fault tolerance, self-healing, and other reliability mechanisms to account for run time faults and exceptions, and to allow systems to operate robustly. These important techniques are demonstrated to enhance the reliability of WSN systems, but given the error-prone nature of WSNs, even a highly reliable system using these techniques cannot be guaranteed to have no failures over its lifetime. Even if the failures are rare, they can still cause serious problems and make the system unacceptable for mission-critical applications. Thus, although it is important, studying how to improve a system's reliability is not sufficient. *A crucial problem for mission-critical WSN systems is how to promptly find that a system cannot provide its important services at runtime.* If a system has such an assurance capability, its administrators can be notified when a failure occurs and can start repairing it. In the mean time, backup systems can be used to perform the system's critical services. Thus, given a RTA capability, even unreliable WSN systems can be trusted to run critical applications.

Therefore, we deem this assurance capability as an important requisite of WSN systems, and propose a novel RTA design principle that requires that systems can be periodically validated to demonstrate their run time operability and the integrity of their application semantics. The essence of run time assurance (RTA) lies in the system's capability to identify failures at the application-level on a periodical basis or by request. If a system has such an assurance capability, system administrators can utilize that to evaluate the system and make decisions whether it is necessary to take actions to recover the system. We believe that RTA can be used to provide application-level reliabilities, and should be addressed as a first design principle for mission-critical systems.

RTA differs from network health monitoring, which detects and reports low-level hardware faults, such as node or route failures [82, 86]. The end-to-end application-level tests that are used for RTA have two key advantages over the tests of individual hardware components used for health monitoring: 1) fewer false positives; RTA does not test nodes, logic, or wireless links that are not necessary for correct system operation, and therefore produces fewer maintenance dispatches than health monitoring systems 2) fewer false negatives; a network health monitoring system will only validate that all nodes are alive and have a route to a base station, but does not test more subtle causes of failure such as topological changes or clock drift. In contrast, the RTA approach tests all ways that an application may fail because it uses end-to-end tests. Network health monitoring improves system reliability by detecting some types of failures, but stops short of actually validating correct system operation. The goal of RTA, instead, is to provide a positive affirmation of correct operation.

The RTA methodology is built around the following three principles:

Run time verification: The RTA principle requires that a system demonstrates at run time that it is able to perform its key services. Testing and debugging techniques are used to fully test the system at design time, and deployment-time validation [65] is used to verify the system during deployment. However, due to the changing environment and the dynamic nature of failures, we argue that even when these techniques have been employed, RTA is still necessary.

Application-level guarantees: Many WSNs are complicated systems, consisting of numerous components and protocols. Each component may use various fault tolerance, self-healing, or other reliability mechanisms to operate robustly. However, even if one can guarantee that each separate component works correctly, the system may still fail to perform some high-level operations. And a user (such as a fire inspector) is only concerned with whether the system performs the way they want, rather than whether each component works correctly. The goal of the RTA principle is to address this and focus on verifying the application-level services.

Correctness demonstration: There are different ways to demonstrate services at run time. One option is to monitor system health information and infer system correctness from this information. Such health monitoring techniques are shown to be effective for certain applications with regular traffic [7,29,69]. However, many critical events, such as fire or volcano eruptions, are rare. Also, in complex WSN systems, it is hard to determine which states should be monitored and how to infer the correctness of services. Monitoring too many states is inefficient and may cause many false positives, but monitoring too few states could fail to reveal failures. For example, Memento [86] uses a periodic heart beat method to determine if a node is still alive. This approach is not suitable for RTA for two main reasons: 1) node responsiveness does not imply proper functioning of the system and its application semantics; 2) node failure does not imply the failure of the overall system or the application. If we have some level of node redundancy, a small number of node failures may not affect the system application at all. Thus, in general, health monitoring techniques are not sufficient to provide confidence in application-level requirements for WSNs at run time. Therefore, we employ testing to verify the proper operation of the application. Using tests allows us to check if the application provides the results we expect regardless of what the state of its components is.

We propose and demonstrate a RTA methodology, in which we validate at run time that a WSN will function correctly, irrespective of any changes to operating conditions since it was originally designed and deployed. The basic approach is to use program analysis and compiler techniques to automate WSN code generations and facilitate automated testing of a WSN at run time. The developer specifies the application using a high-level specification, which is compiled into both (i) the code that will execute the application on the WSN, and (ii) a set of input/output tests that can be used to verify correct operation of the application. The test inputs are then supplied to the WSN at run time, either periodically or by request. The WSN performs all computations, message passing, and other distributed operations required to produce output values and actions, which are compared to the expected outputs. This testing process produces an end-to-end application level validation of all components required for correct system operation, including hardware, software, network topology, and any interactions between them. The RTA facilities can also collect useful information during the tests and to help identify the root of a failure and repair the system.

5.2 Implementation framework

We have implemented a framework for designing and automatically testing WSN applications using the RTA methodology. In our framework, the developer first specifies the application using a high-level Sensor Network Event Description Language (SNEDL) [50], which is an extended Petri net model. Second, we develop a code generation tool to compiles the SNEDL model down to TinyOS [61] code that runs on the Telos nodes [24], as well as tests the defined mappings between sensor input values and system outputs. The code generation explicitly encodes the high-level logic of the system into the arc-transition-place pattern of the SNEDL model. The system executions in this architecture are driven by tokens, which travel between places through transitions via arcs. The architecture naturally supports RTA. We can run various tests by simply injecting testing tokens. We use program analysis techniques to identify the minimal set of tests that will cover all system components used by the application, such as sensors, nodes, links, and application logic. This anal-

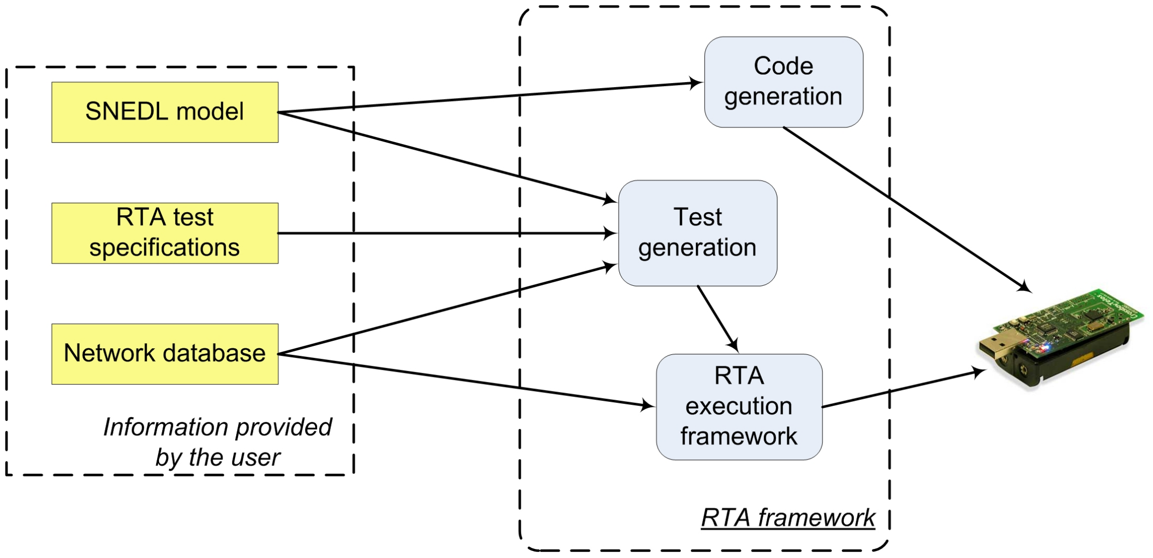


Figure 5.1: Run time assurance framework.

ysis uses new techniques that exploit information about network topology and the redundancy of nodes based on sensing range, and builds on existing techniques to cover all execution paths in the program [54]. Once the code for the sensors has been generated and deployed, and the proper set of tests has been created, the RTA execution mechanism can start running the RTA tests on the network and thus monitor the system's functionality. The interactions between the components of the RTA framework are shown on Figure 5.1.

5.2.1 Application-level Specification

To allow designers to describe the application-level requirements of their system, we need a suitable specification language. Such a language should be able to meet a number of requirements:

- It should be able to capture the state and behaviour of the system and support key features of WSNs such as different sensor types.

- Since RTA is primarily concerned with application-level requirements, the specification should be able to operate at that level.

- It needs to be clear and simple to use since it should be able to act as an interface between people who register events (e.g. application semantics experts) and the sensor network designers.

There are some additional benefits to using a well defined specification language. The first one is that a specification could serve as a guideline to the programmers. A clear and detailed specification can easily be followed when developing the RTA part of the application. The second benefit is that such a specification can be used to automatically generate parts of the RTA code. This could help decrease both the development time and the number of programming errors.

5.2.1.1 The SNEDL model

We decided to use SNEDL [50] for an underlying RTA specification language. SNEDL is the first event specification language to support key features of WSNs. As a description language, it is an extension of Petri nets. A basic Petri net consists of places (circles), transitions (rectangles or bars), directed arcs, and tokens (dots inside places). Transitions model various kinds of actions, tokens model instances/ objects, and places represent the states in which the objects can be. Arcs represent the way in which objects are created or destroyed; they also represent changes between states [35].

An alternative to using SNEDL for event description is employing SQL or SQL-like semantics. However, Franklin [67] points out that SQL-like semantics are not always suitable for sensor networks because of the lack of collaborative decision making and other necessary features. Admittedly, SNEDL cannot capture the lower-level logic of a WNS system but since RTA works at the higher level, the logic provided by SNEDL matches RTA's requirements.

Figure 5.2 shows an example SNEDL model of an application that monitors the temperature and humidity values in the area where the sensors are deployed and signals if these values are above (below) some predefined thresholds. The SNEDL model consists of places (circles), transitions (rectangles or bars), directed arcs, and tokens (dots inside places).

- Places represent the states in which the objects can be. In SNEDL, dashed places are used to abstract sensors events, i.e. sensors (places 1 and 3). These dashed places are where tokens representing the physical sensor readings are generated. Higher level events are constructed using the sensor events.

- Transitions model various kinds of actions. They represent the decision part of the application - they check for the occurrence of specific conditions and determine the steps to be taken when

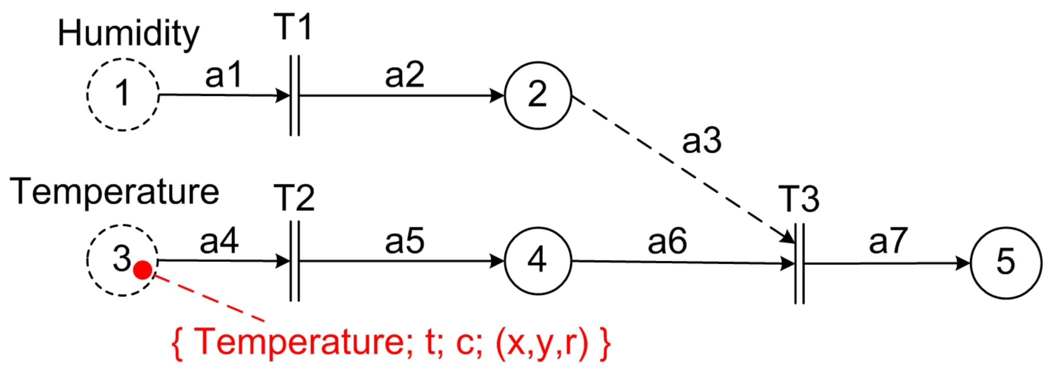


Figure 5.2: An example SNEDL model.

these conditions are satisfied.

- In a traditional Petri net arcs represent the way in which objects are created or destroyed as well as changes between states [35]. There are two types of arcs in SNEDL: general arcs and radio links. A radio link, shown as a dashed line (arc a3), denotes communication among nodes. In our example model, in order for transition T3 to fire, a node needs to detect sensor event 3 and also receive a message from another node that sensor event 1 has been detected.

- Tokens model instances/objects. The tokens that arrive at each sensor event are associated with temporal and spatial attributes, and therefore the information about when and where the data has been sensed can be retrieved. For example, if a token with a time stamp t, capacity c, and location attribute (x, y, r) reaches a temperature sensor event (place 3), we can say that a temperature sensor at location (x,y) with sensing range r has a value c at time t.

The SNEDL model of the WSN application is one of the inputs to our RTA framework. The user is asked to write a script to specify the transitions and places of their model as well as the connections among them. We provide a set of predefined transition types which we believe to cover the majority of logical operations used in a WSN application, such as, greater, less, equal, minimum, maximum, and difference between two consecutive values, average, and moving average. If, however, the user wants to specify transitions with more complex functionality, they are given the option to do so.

An advantage of SNEDL is that it provides us with a different perspective of a WSN system. By representing a WSN system as a SNEDL model we can view it as a flow of tokens from places in the

Petri net to other places. There are several reasons why this is useful for the RTA specification. First, this is very suitable for testing purposes and makes running RTA tests extremely straightforward. Second, a token-flow model allows us to easily differentiate between a real event and a test event. Since the test events are specified by test-tokens, all we have to do is mark the test-tokens as such. Then, whenever a transition is triggered or a place is reached, we can always check the type of the token that caused this to happen and react accordingly. Third, such a model helps the collection of event-logs and makes it easy to define flow-traces in the system.

We present an example RTA test to support our claim that SNEDL makes RTA testing straightforward and easy. We want to test if the previously described system properly detects the occurrence of a fire. The test should be run in every room where these abnormal signal may be encountered. To test a single room we need two tokens: $token_hum = (RTATestID, v1, t1, room1)$ and $token_temp = (RTATestID, v2, t2, room1)$, where *v1* and *v2* are the token values which in this case will be "true" to show that the abnormal humidity and temperature have been detected; *t1* and *t2* are the generation times of the tokens; and *room1* is the room for the first test. The inputs to our test will be $input1 = (token_hum, place_1)$ and $input2 = (token_temp, place_3)$. Both inputs specify a tuple of a token and the place to which the token should be injected.

The result we expect to see is an evacuation alarm no longer than 15 seconds after the two phenomena have been detected. Therefore, the output of the test is $output1 = (place_5, 15sec, room1)$. Now that we have the inputs and the outputs, we can define the whole test.

$test1 = \{RTATestID, \{input1, input2\}, output1\}$

Running the test for the rest of the factory rooms only requires changing the location in the test-tokens.

5.2.1.2 RTA specification

The SNEDL can specify the application-level requirements and logics, and it also makes easy to automate test generation. This test generation needs users to input the requirement specification for RTA tests. The user should provide information about what events they want tests generated for, the parts of the network they want to test, and at what times the test should be run. An example of such a test specification is:

```
//equivalence -
//'no equivalence' or 'region equivalence'
region equivalence

//Declare basic elements of the language
Time  T1, T2, T3, T4;
Region R1, R2, R3;
Event EFire;

//Define the elements by annotation
T1=07:05:00, 10/15/2009;
T2=07:10:00, 10/15/2009;
T3=07:12:00, 10/15/2009
T4=07:15:00, 10/15/2009;

R1={Room1 | node1, node2, node3};
R2={Room24 | node4, node6};
R3={Room15};

EFire = Fire @ (T1 to T4);
```

5.2.2 Automated Code Generation

Once the designers have the SNEDL model to describe their application, the next step is to generate the code with RTA features. The code should not only implement the application logic according to the SNEDL model, but also provide facilities to run RTA tests at runtime.

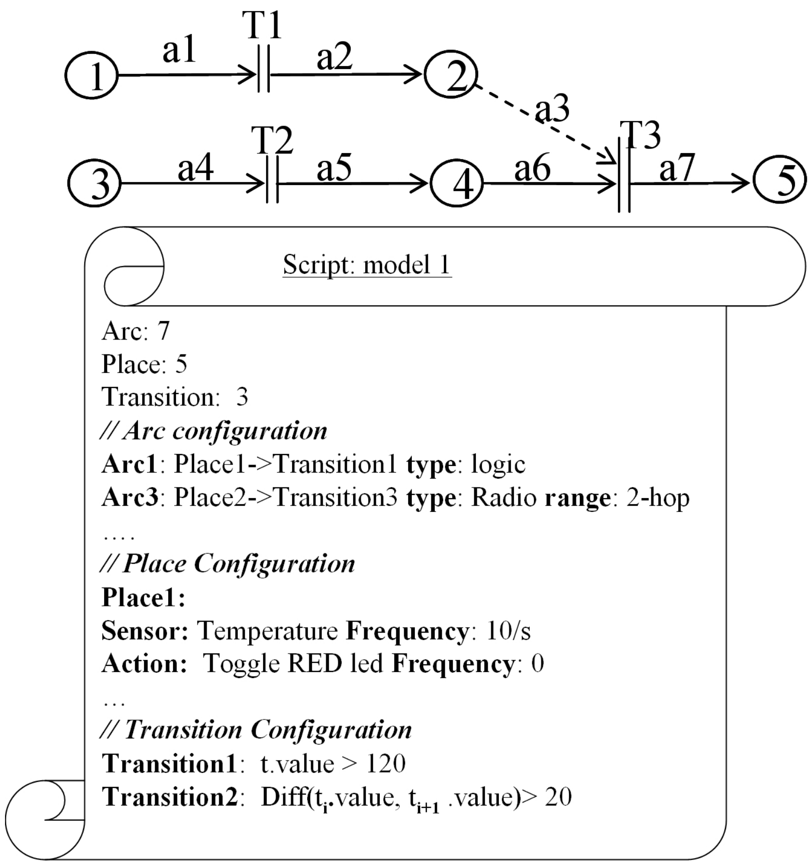
One potential way to implement the RTA facilities is to use annotations and a preprocessing step. Users only need to annotate the input and output places of the tests, and then the preprocessor is responsible for automatically generating the code and integrating the modules to run it. A similar method, allowing users to specify which data they want logged, is used in Envirolog [66].

However, this annotation technique is not suitable for implementing RTA facilities. First, in order to support all potential tests, users must identify all possible test entries and exits in the program, and manually translate the RTA requirements into certain program entries and exits. This process is time-consuming and error-prone. In addition, adding extra code at all places significantly increases the size of the executables. Further, this annotation technique cannot solve the conflict between tests and real events. After a test is triggered and the system starts executing, one cannot distinguish between a test and real event execution.

Another approach is to use the RPC facilities provided by the Marionette [31] tool suite. With Marionette, users can trigger certain tests without any extra coding by remotely calling functions on the nodes. However, this method cannot address all of our design goals. First, not all tests are triggered by function calls. Some test inputs should be injected in the middle of a function which RPC cannot provide. Consider, for example, a function with a long chain of operations. We cannot use an RPC to inject a test input in the middle of this operation chain. Second, this method also cannot deal with the conflict between real events and tests. Third, it can only support virtual event generators, but cannot provide record and replay events or log collection.

Since the above methods cannot fully support all RTA features, we develop an Automatic Code Generator (ACG) which accepts a SNEDL model as input and produces TinyOS code [1]. This approach eases the process of writing TinyOS code and improves code correctness. More importantly, the TinyOS code generated conforms to a SNEDL structure, where we explicitly implement the SNEDL logic object (Place and Transition) as tinyOS components, and then the code execution is driven by tokens traversing through SNEDL objects. The code with such a structure can 1) receive new test inputs, automatically run the tests and verify the test results conform to the specification; 2) collect trace and other information during tests; 3) differentiate executions of real events and tests, and solve potential conflicts among them.

[1] In this paper, we focus on generating TinyOS code, but our approach can be adapted to generate code in other languages.



Script: model 1

Arc: 7
Place: 5
Transition: 3
// Arc configuration
Arc1: Place1->Transition1 **type**: logic
Arc3: Place2->Transition3 **type**: Radio **range**: 2-hop
....
// Place Configuration
Place1:
Sensor: Temperature **Frequency**: 10/s
Action: Toggle RED led **Frequency**: 0
...
// Transition Configuration
Transition1: t.value > 120
Transition2: Diff(t_i.value, t_{i+1} .value)> 20

Figure 5.3: Script to specify a SNEDL model

5.2.2.1 Input and Partition

The input of our ACG is a script file that designers write to specify the SNEDL model. An example is shown in Figure 5.3. Users first assign an ID to each SNEDL object, and then input the connections of the model by configuring arcs' source and destination. In the SNEDL model, arcs have three types: logic, radio and hybrid. Logic arcs connect places and transitions in the same node. Radio arcs, shown as dashed lines, indicate the token flow across nodes. For example, radio arc a3 in Figure 5.3 represent that when a temperature sensor gets a reading greater than 120 (Transition 1), it arrives the state place2, and sends a token to nodes within 2-hop range. Hybrid arcs indicate that a transition would receive multiple tokens from its own places and other nodes. One example of hybrid arcs in shown in Figure 5.7.

The script also specifies places in the SNEDL model. Places represent execution states that nodes may reach at runtime. Every place may be associated with two attributes: sensors and actions. At some places, nodes need to access sensors to get data, such as place1 and place3 in Figure 5.3. In the script, designers specify sensor types and sampling frequencies for these places. According to the SNEDL model, each place has at most one sensor. On the other side, one place may have multiple actions, which are operations that nodes can do after reaching certain states, such as beep, turn on/off LEDs and report messages to sinks. At last, every transition logic is specified with logic primitives described in Section 5.2.1.

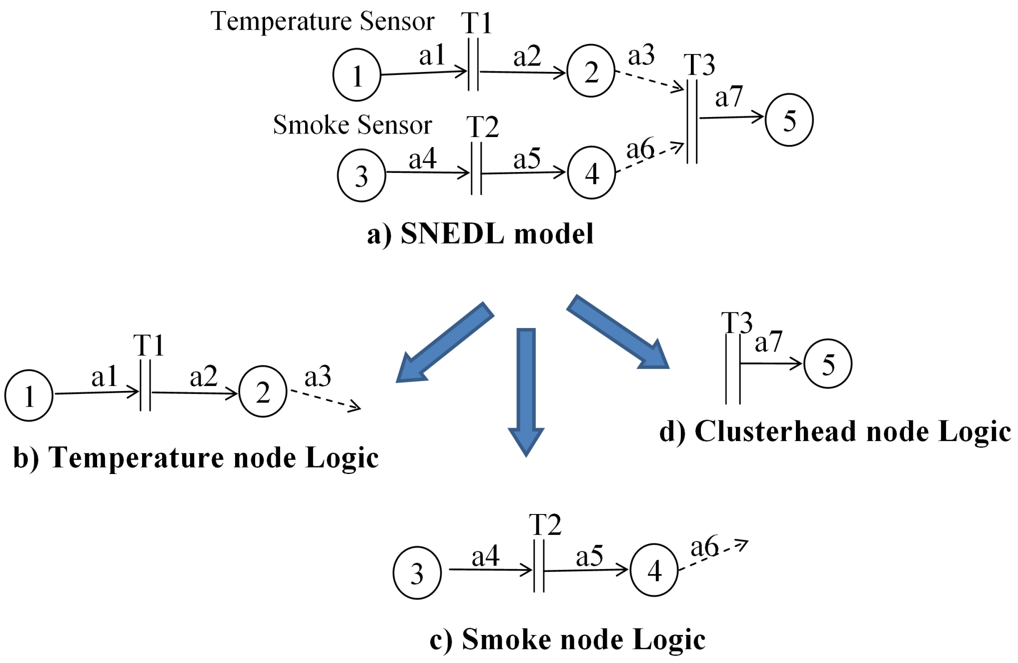


Figure 5.4: Partition a SNEDL model

Before ACG generates TinyOS code, it needs to partition the SNEDL model into different programs. Nodes in heterogeneous WSNs have different functionalities and should run different code. Consider a fire detection system that consists of smoke sensor node, temperature node, and cluster-head node. Smoke and temperature nodes sense and notify cluster-heads if detecting abnormal readings. Cluster-head nodes collect information from both sensors and have logic to detect fires. The SNEDL model is shown in Figure 5.4. It is not efficient to deploy the same copy of code on all three types of nodes. Our ACG is designed to generate different code for each node that only implement its only application logic. The ACG first automatically partitions the SNEDL model into three parts, and then generates one program for each part. To do the partition, we first construct a directional graph with places and transitions as graph nodes, and logic and hybrid arcs as edges. Then to partition the SNEDL model is actually to find connected components in this graph. Our ACG applies a breadth-first-search to find the connected components.

5.2.2.2 Code Structure

TinyOS programs are built out of software components. In order to translate the SNEDL model into tinyOS code, we add a SNEDL structure above the tinyOS component model. This structure consists of components that represent places and transitions. Using this structure, code executions are driven, processed, represented and recorded as tokens traversing through places and transitions.

The core of our code structure is the concept of *token*. A token is defined as an encapsulation of a value, temporal and spatial information, and a RTA ID. The value of a token may have different meanings at different times. At the early stages of the system executions the value is usually a sensor reading. Later, it may have some semantic meaning. For example, in a pollution control system, the value of a token can represent different types of pollutions. The time and location attributes of a token represent when and where this token was generated. We are using absolute time[2], and the location can be defined as Latitude and Longitude, or in a semantic way (e.g. room numbers). The RTA ID attribute shows which RTA tests this token belongs to. We use ID 0 to indicate the real execution. For a test, tokens of this test use the test ID as the RTA ID. Tokens of one execution always have the same RTA ID. This allows us to differentiate tests and real events by just looking at token IDs.

Tokens trigger the system execution and traverse across components to transfer information. Usually, tokens are firstly generated by sensors, and then are inputted into the place component. At a place, tokens trigger actions of the place. When reaching a transition, the transition logic decides whether to pass this token. By recording the traversing history of tokens, we can continuously monitor system execution, and collect traces of system execution.

[2]We assume that a system with RTA must have facilities to keep every node time-synchronized. For most mission-critical applications, time synchronization is a must-have condition.

In our code structure, the most important component, called *SNEDL_ResolverM*, works as a system engine. Its first task is to create arc, places and transition objects and sets up their connections according to SNEDL models. The ACG generates an initialization function for this purpose. Each logic objects will be assigned with one unique ID, such that they can be accessed with one common interface. In this component, each logic object has a mode attribute which provides three options: transfer, stop, and log. The transfer mode is the default model. Under the log mode, objects record all tokens passing them, and objects under the stop mode block tokens. The mode attribute of any object can be configured at the runtime.

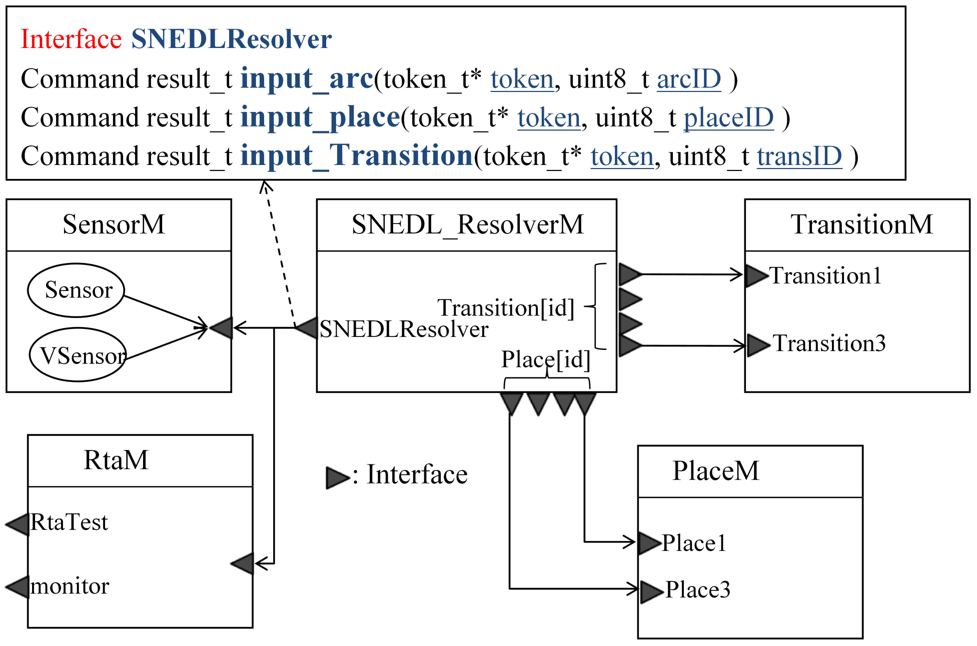
Next, we generate two tinyOS components: PlaceM and TransitionM. PlaceM is a container of all place actions. For each place in the SNEDL model, the ACG generates an event function *place.action*, which is called to perform corresponding actions when a token reach this place. Similarly, TransitionM contains all transition event functions, which are generated by the ACG. Each transition event function receives tokens, and call a command to return a new token if its logic is satisfied.

```
// PlaceM Component
...
event token_t* Place3.action(token_t* token){
   call Leds.redToggle();
   // report this token to the sink.
   call Report.Send(token);
   return token;
}

// TransitionM Component
...
event token_t* Transition3.receive(token_t* token){
   if (token == NULL) return NULL;
   if (token->value > 120)
   call Transition3.returnToken(token);
   return token;
```

}

Figure 5.5 shows the component structure of the code generated. The SensorM component contains all sensors in the SNEDL model. The ACG generates the code to sample sensors, packs the reading with location and time information into a token, and inputs the token into *SNEDL_ResolverM* to trigger the execution. *SNEDL_ResolverM* accepts tokens as inputs and then automatically sends them to the next logical state. When a token is received at some arc, *SNEDL_ResolverM* stops, runs or logs the token according to the status of the arc, and then finds the transition that this arc connects to. Next, it dispatches a *Transition.receive* event through the *Transition* interface, which is a parametrized interface that can dispatch *Transition.receive* events to corresponding transition functions. If the token is passed, a transition calls the *passToken* command to give a new token back to *SNEDL_ResolverM*. On receiving a token from a transition, *SNEDL_ResolverM* finds the places following the transition and generates *place.action* events through the *Place* interface.



```
Interface SNEDLResolver
Command result_t input_arc(token_t* token, uint8_t arcID )
Command result_t input_place(token_t* token, uint8_t placeID )
Command result_t input_Transition(token_t* token, uint8_t transID )
```

Figure 5.5: Component Structure

5.2.2.3 Test Component

The major testing component is *RtaM* component, which provides more test facilities. *RtaM* keeps a test list. Its major task is to receive test configuration messages from sinks, inject the test inputs into the system at the right time, and to check whether the token reaches the required places after the deadline. It also provides a *RtaTest* interface that other components can use to trigger tests starting from any arc/place/transition. Furthermore, it has a *monitor* interface which can be used to get logs from any logic object.

The *RtaM* component can support various types of tests:

- Test sent from sinks. Users can use this ability to change their tests at run time, and have an adaptive testing plan to verify the reliability under different scenarios.

- Record and replay. Nodes can record the logs from previous real events, and replay them for testing. This feature is implemented using the *Run_log* feature of arcs. We use the *config* command of the *SnedlResolver* to set the status of these arcs to *Run_log*. When real events take place, the arcs record the tokens into the nodes' flash memory. To replay an event, the token logs are read from the flash, the *RtaM* component set the RTA IDs to a value different from 0, and input them to the execution flow.

- Partial tests. Even though RTA requires end-to-end tests in most cases, partial tests may be needed to identify root causes if the e2e verification fails. With the *RtaTest* interface, users can specify tests starting and ending at any arc/place/transition.

Our code structure also provides flexible ways to deal with the collisions of real and test events. Real events and tests use tokens with different RTA IDs. Every arc, transition, and place can differentiate between them. One simple policy is to always cancel the test tokens in the presence of collisions. However, this policy might not always be suitable. For example, suppose that a fire detection system also periodically collects temperature readings. If the test-cancel policy is applied, the tokens of this collection routine will always cancel the test tokens. Therefore, a better policy is to use different actions to handle conflicts at different stages. Test tokens and real tokens can co-exist in the early stages of system execution, given that the test tokens do not affect the real ones. On the other hand, for transitions and places that require urgent actions, the real tokens should always cancel the test tokens.

In summary: 1) this code structure provide rich testing functionalities. It efficiently supports self-testing at run time. We can either do a end-to-end test by using virtual sensors, or a partial test by injecting tokens directly into any arc/place/transition; 2) it is easy to monitor execution states and collect running traces by using the log mode; 3) when real events and tests occurs at the same time, we can use the tokens' RTA IDs to distinguish between them, and cancel tests by blocking only test tokens.

The ACG can also be adapted to generate the code in other languages. For example, we can build specific structures or classes for different SNDEL objects, and change the code templates of the ACG to generate the code in these languages.

5.2.3 Automated Test Generation

One of the benefits of our RTA framework is that it comes with an automatic test generator. Our automatic test generation takes three inputs:

1. the SNEDL model of the application. Since the application code is automatically generated based on the SNEDL model, we analyze the model itself.
2. the network topology;
3. the test specification. The user should provide information about what events they want tests generated for, the parts of the network they want to test, and at what times the test should be run. An example of such a test specification is shown in Section 5.2.1.2.

We propose three reduction steps that help us decrease this huge number of test inputs. The first step has been widely used for reducing the number of tests for software applications. The second and third steps, however, are novel and also unique to wireless sensor applications.

This part of work was mainly done by Ms. Krasimira Kapitanova. Interested Reader can refer to Section 4.2 in [105].

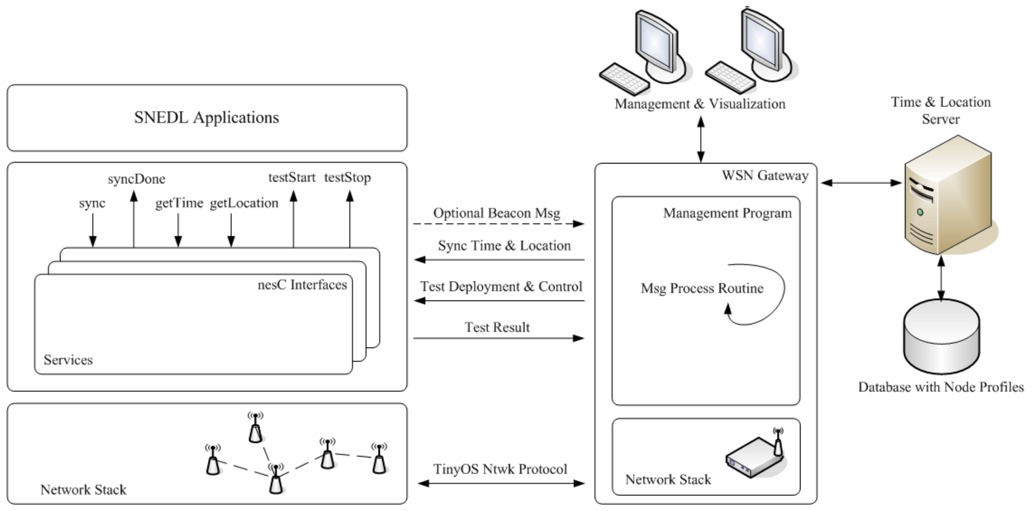


Figure 5.6: RTA execution framework

5.2.4 Test Execution Support

Test execution support runs the RTA tests on real networked sensing devices. Its execution flow is shown in Figure 5.6. It involves the following steps:

1. After system initialization, every node in the sensor network uses the ”sync” interface provided by the service layer to request time synchronization with the WSN gateway and request its own location, the gateway will get synchronized with the time and location server and then process the request from each node, the location server can get node locations by queries the database. After the synchronization step, each node knows the current time and its own location.

2. Users can schedule RTA test by writing a test specification at user terminal, the ATG program will then parse the user's specification and analyze the SNEDL model, and then generate virtual sensor readings that will correspond to events in the system (e.g., a fire event) for each node.

3. The virtual sensor readings (i.e., the test data) are then passed to the gateway program, and shipped to each node automatically using the test deployment protocol, and each test is scheduled to automatically start and stop at the specified time using the test control protocol.

4. When the scheduled test start time is reached, each sensor node will start readings from its virtual sensors. The virtual sensor readings will then create virtual events in the network, these events will be detected and processed by the SNEDL logic at the application layer.

5. When virtual events are detected by the SNEDL, these events will be reported to the gateway, and then passed to user terminals for resolution.

Since the RTA test process is completely automated, users only need to write the test specification once to schedule tests. The test specification will be processed by the ATG program to generate test data automatically, the test data will then be deployed to the network automatically using the test execution support. When scheduled test time arrives, sensor nodes will read from virtual sensors and simulate virtual events, these events will get processed by application logic, and application layer report will be sent back via the test execution support to user terminals, users will eventually get feedbacks to see whether the system has passed the RTA test. Thus, the entire process is automated, with minimum user attendance at run-time.

With support of the ACG, ATG and the execution framework, the RTA methodology can be applied to different user applications with minimal effort. Users can first specify their own application using SNEDL, and then ACG will generate code for sensor nodes automatically, reflecting changes in the application logic. On the other hand, ATG will use the same SNEDL model, and generate RTA tests according to user specifications. These tests will be automatically deployed and run on the sensor networks. Users will eventually get automatic updates on RTA test results.

5.3 Case Study

We present a case study to demonstrate the usability of our RTA methodology. In this scenario, we are required to design and build a fire detection system for a building. The building has seven floors and there are ten rooms per floor. There is at least one node with a smoke sensor and one node with a temperature sensor in each room. The fire detection application uses both smoke and temperature readings to determine the presence of fire. The fire detection algorithm we use is composed of two separate algorithms. The first monitors the temperature and smoke increase rates and if they both exceed the predefined thresholds, the algorithm reports the presence of fire. The second algorithm reports fire if the temperature value exceeds a predefined threshold.

Based on this description, we generate the SNEDL model that represents the application's logic, shown in Figure 5.7. The elements of this model are:

- Places S_1 and S_2 represent the two types of sensor events, while LA_s, LA_t, and Fire stand for, Local smoke alarm, Local temperature alarm, and Fire, respectively.

- Transitions: Transition T_{DS} is fired if the smoke increase rate goes above a specific predefined values. This causes a message to be sent over the radio (represented by a dashed line in the figure) and a Local smoke alarm to be raised. Transition T_{DT} fires if the temperature increase ratio goes above another predefined threshold. This also leads to sending a message to the radio and raising a Local temperature alarm. Transition T_{FIRE} is fired if fire has been detected. It is activated only if there are more than two tokens ready to enter it.

- For T_{FIRE} we require that the tokens were generated from sensor readings from the same room.

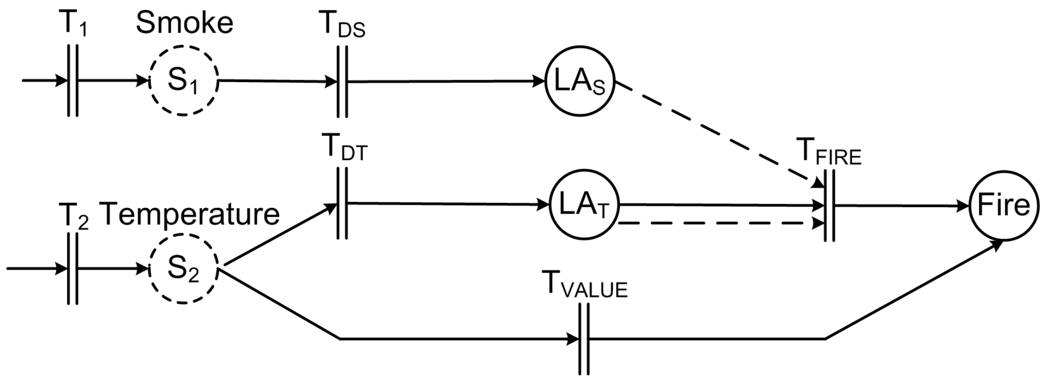


Figure 5.7: Fire detection SNEDL model.

Next, we implement the system. First, we write a new script file according to this SNEDL model, and input it into the ACG. The ACG partitions the whole logic and generates the code for the smoke and temperature sensors. The code structure is described in Section 5.2.2.2.

To automatically generate the test suite, we need the user to provide us with an RTA test specification. The testing specification in Section 3.2 could be used for our scenario. It contains information about which regions we have to test (rooms 1, 15, and 24) and at what times (the morning of October 15th). As indicated by the specification, we are interested in event EFire. The tests we run for this scenario provide inputs for places S_1 and S_2 such that the presence of fire is simulated. The WSN system is expected to report the presence of fire based on the virtual readings from the tests. Knowing the test specification, the SNEDL model, and the topology, we automatically generate the necessary set of tests to be run by the execution support.

This case study shows how the RTA methodology is used to guide and help system designers build a fire detection system with RTA capabilities. As demonstrated by this example, the formal SNEDL model and the RTA specification language make the system logic and the RTA requirements clear, unambiguous, and easy to understand. Our ACG, combined with the SNEDL model, efficiently helps designers generate an accurate logical implementation of a system's high-level behaviors. Most importantly, our methodology fully addresses the RTA requirements.

At first this case study might seem simplistic. However, several key points must be emphasized. It is important to recognize that this simple model can represent a fire detection system that exists across many floors and uses many sensors. Although we had initially specified that we have seven floors and seventy rooms, these numbers do not confine the application model. The same model could successfully be used to detect fires in a skyscraper with hundreds of rooms. In other words, even though the model is simple, it can represent a large scale system. This case study also uses relatively simple logic for temperature and smoke sensors. However, the power of the underlying Petri net allows us to describe arbitrarily complex logic and control flow. An additional advantage is that the resulting model of a complex application is not necessarily too big or knotty since much of the complexity for such systems is encompassed in the logic associated with the transitions.

5.4 Evaluation

In order to investigate the performance of our RTA methodology, we implement a prototype Fire Detection (FD) system according to the SNEDL model in the case study. The system is built on an indoor testbed. The testbed is composed of 21 TelosB nodes, placed in a 7×3 grid on a board. One node is chosen as the base station and the rest are divided into different rooms. Since every node is only equipped with a light sensor, we use light sensors to simulate the temperature sensors.

We use the fire detection algorithm described in Section 5.3. However, the nodes only contain the logic for the temperature sensors. Once a node detects a fire, it sends a report to the base station. For the communication topology we constrain the nodes to only directly communicate with nodes in the same room. For multi-hop communication we use a simple geographic forwarding routing protocol.

5.4.1 Robustness to Failure

The goal of RTA is to maintain the accurate operation of a WSN application, despite the unreliable and failure-prone underlying infrastructure. In order to evaluate the robustness of our RTA approach, we have compared it to a health monitoring (HM) system and a pure system with no RTA or health monitoring support. All three systems execute a FD application with the same logic. The RTA FD system is generated with the help of our RTA framework. We test all rooms on a rotation basis and a different room is picked each time a test should be run. The FD system with HM monitoring is implemented following the HM mechanism introduced by Memento [86]. Memento uses heartbeats from neighbor nodes to determine if a node is functional. We assume that when an RTA test fails or the HM system detects a node failure the failed nodes are immediately repaired.

5.4.1.1 Experimental Results

To demonstrate that RTA is effective in maintaining application robustness while significantly reducing maintenance cost, we have run multiple experiments under different settings. We use a flashlight to generate the fire events in our testbed. To measure the system's performance under realistic scenarios, the fire event sequences are generated using a Poisson process generator. The room in which a fire occurs is chosen randomly. We also compute a failure sequence by the random Poisson process to determine which nodes will stop execution at what time. During experiments, the WSN gateway injects node-failures by sending messages to specific nodes and stops their execution strictly according to the failure sequence. For each experiment, we run the three systems sequentially with the same fire and failure sequences. A node is periodically chosen to fail until no operational nodes are left and the experiment stops. To represent the robustness of the system

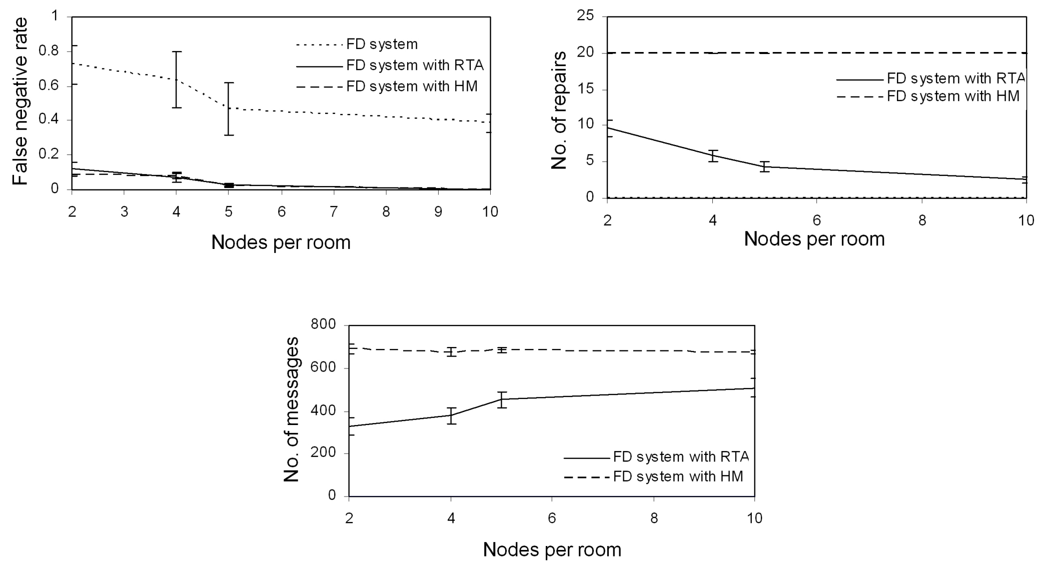


Figure 5.8: System robustness and maintenance cost with different level of redundancy: (a) RTA and HM achieve similar false negative rates; (b) RTA requires 50%-70% fewer repairs than HM; (c) On average, RTA uses 33% less messages than HM.

we use the false negative rate which we define as the ratio between the number of unreported fires and the total number of generated fires. Maintenance cost is measured with the number of repairs, the Mean Time of Repairs (MTR), i.e. the average time between two consecutive repairs, and the number of messages sent during testing.

In the first set of experiments we measure the robustness and maintenance costs of the three systems. To study the impact of redundancy, we vary the number of nodes per room from 2 to 10. All experiments last 20 minutes and both the RTA and the HM system run a test every minute. The fire event rate is 0.5/s, and the node failure rate is 1/s. Figure 5.8a) shows the results of these experiments. Each data point is the average of 5 trials with a 90% confidence interval. The results reveal that, compared to a pure system, both RTA and HM significantly reduce the false negative rates of the FD application. This is expected as both mechanisms can detect node failures and repair nodes immediately. Comparing the RTA system to the HM system shows that both systems demonstrate similar robustness levels. The only visible difference is when there are only 2 nodes per room. Since according to the application logic we need at least two nodes per room, any node failure will lead to a false negative. In this case, the false negative rate of the RTA system is slightly greater than that of the HM system because the RTA system only tests one room at a time while the HM system tests all nodes every time. Increasing the level of redundancy eliminates this difference in the false negative rates. With 5 nodes per room, both systems have a false negative rate close to zero.

We also compare the maintenance cost of the RTA and HM systems. Figure 5.8b) shows the number of repairs of both systems. We can see that the RTA system requires much less repair than the HM system. The number of repairs required by the HM system is constant since it triggers a repair every time a node fails. On the other hand, the RTA system repairs nodes only when the live nodes in a room cannot detect the fire. Our results show that with a certain level of redundancy, the RTA mechanism can significantly reduce the number of repairs while still providing high confidence. For example, with 10 nodes per room, the RTA system guarantees no false negatives and requires only a total of 2.5 repairs. On average, the RTA system produces 70% fewer maintenance dispatches. We also measure the MTR of both systems. Results show that the RTA system has much longer MTR than the HM system. For example, with 5 nodes per room, the MTR of the RTA system is 5.6 seconds, with 1.5 seconds for the HM system.

Another type of maintenance cost is the extra messages incurred by running tests. The base station in the RTA system needs to send messages to trigger tests and nodes are required to report virtual fire events during tests. For the HM system, nodes need to send heartbeats, and the base station should be notified if no heartbeat is received from some node. These extra messages use bandwidth, consume extra energy, and reduce the system's lifetime. We have measured the number of the extra messages for both systems. Results are shown in Figure 5.8c). We can see that the RTA system introduces much less message overhead. The HM system always uses over 650 messages, because nodes keep sending heartbeats. On average, the RTA system uses 33% less messages than the HM system.

Node level failures are lower level failures and the HM mechanism has been specifically designed to detect them. In the next experiment we compare the performance of the two systems when application level failures are introduced. To do this we insert a new type of error, which we call *location error*. A location error can be defined as any unexpected changes in the location of a node from one region/room to another. Such a location change should lead to a failure if it is against the application's requirements. In our scenario, we need to have at least two nodes per room. Therefore, a location change that decreases the number of nodes in a room to less than two should cause an application-level failure.

In this experiment, we change the location of nodes to simulate location errors. We still use a Poisson sequence to generate location errors with the rate of 1/s, and randomly chosen nodes change their locations to adjacent rooms. The false negative rates of our three FD systems are shown in Figure 5.9. We see that the HM system has a high false negative rate and cannot guarantee the system's robustness. The reason is that when a node is moved to an adjacent room, it can still send heartbeats to its neighbors, so the HM system considers it operational. However, since the RTA system tests the fire detection ability of each room, it can still catch such errors, without using extra mechanisms to explicitly detect location errors. On average, the RTA system misses 75% fewer system failures. Based on these results we can conclude that, compared to an HM system, an RTA system can provide a better visibility into the application's behavior at run time.

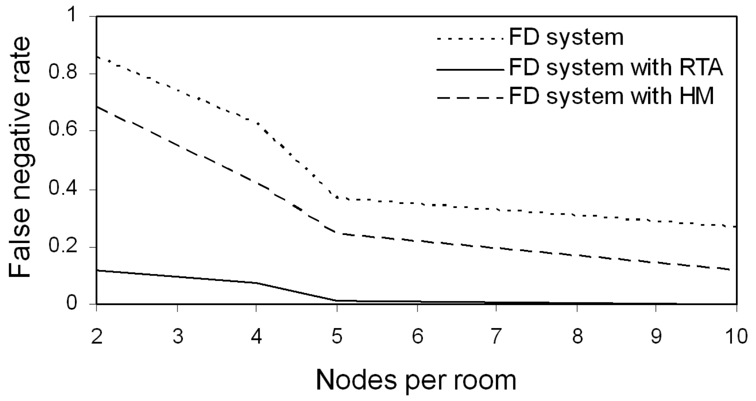


Figure 5.9: With location errors, the RTA system missed 75% fewer fires than the HM system on average.

5.4.1.2 Simulation Results

We have used simulation to demonstrate that RTA significantly reduces the amount of the necessary network maintenance. Our simulation studies a scenario in which we have 25 rooms, and each room has N nodes, where N represents the level of node redundancy in that room. We assume that the lifetime of sensor nodes follows a Poisson distribution. R is the mean lifetime of the sensors nodes in months and it represents the reliability of the nodes. We generate fire events following a Poisson distribution. The expected occurrence interval of these events is 3 months. The simulation period is set to 2 years. Each data point is the average of 5 runs.

We make the following two assumptions when comparing the amount of maintenance work required by RTA and HM: 1) If the RTA tests determine that the nodes in a room have lost the capability of detecting fire, a maintenance worker is dispatched to replace the failed nodes in that room; 2) If the HM determines that a node has failed, a maintenance worker is dispatched to replace that failed node.

In order to achieve a fair comparison, we use the same testing overhead (in terms of the number of testing messages sent) for RTA and HM. On average, an RTA test requires K times more messages than HM - in our implementation K≈8. Therefore, in our simulations RTA runs in 1/8 the frequency of HM. In this simulation the RTA test frequency is a test per day for each room.

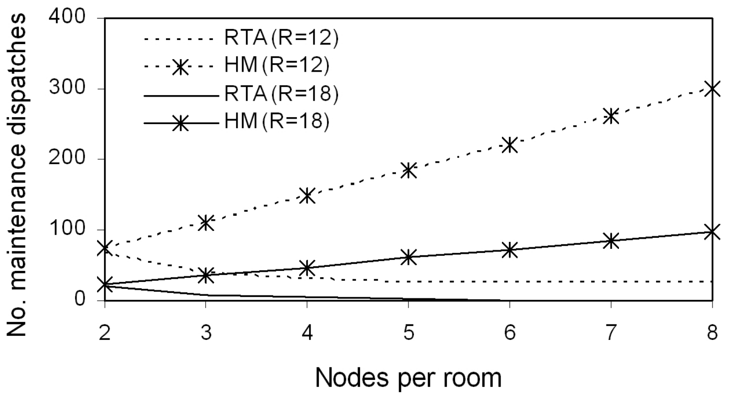


Figure 5.10: RTA reduces the number of maintenance dispatches to only 0.3%-33.9% of HM.

Figure 5.10 shows the number of maintenance dispatches under different levels of node reliability (R) and redundancies (N). We observe that RTA makes a significant reduction in the number of maintenance dispatches (maintenance overhead). Since HM maintains node-level reliability, the number of maintenance dispatches is proportional to the number of node failures. In contrast,

System	ROM (bytes)	RAM (bytes)	lines of code
Pure FD	18142	898	1767
RTA FD	24888	3386	3193
HM FD	22976	1280	2590

Table 5.1: Memory usage and footprint of the three systems.

RTA maintains application-level reliability and a maintenance dispatch is only necessary when the application logic is broken. Thus, by taking advantage of the node redundancy level, RTA can considerably reduce the total number of maintenance dispatches required.

5.4.2 Overhead

In this subsection we study the overhead of our RTA framework. Table 3 compares the memory usage and footprint of the three FD systems. We can see that, because of its token-flow programming structure and more powerful test execution facilities, our RTA system uses more program memory and RAM. Also, the code for our RTA system is larger than that for the other two systems. However, since most of this code is automatically generated by our RTA framework, users only need to write the SNEDL model specification script, which was less than 50 lines for the experimental FD system.

5.5 Conclusions

A major disadvantage of the current reliability and health monitoring techniques used for WSN applications is that they monitor the performance of the low-level components of the system instead of the reliability of the application layer. To the best of our knowledge, this is the first work to address this issue in WSNs and suggest a methodology to help designers and users verify an application's integrity at run time. We have also implemented a framework that facilitates the use of our RTA methodology. This framework provides automated code generation, automated test generation, and execution support for the RTA tests. To decrease the vast number of generated test cases, we introduce three test suite reduction algorithms, two of which are unique to the nature of WSN applications and take advantage of the network's topology and node redundancy.

We evaluated an implementation of a fire detection application with RTA support and found out that RTA performs much better than health monitoring in identifying application-level failures and almost just as good in identifying low-level failures. In addition, our RTA implementation incurs considerably less communication overhead and also decreases the amount of maintenance work that needs to be done in order to keep the system operational.

Chapter 6

Conclusions and Future Work

To conclude this dissertation, in section 6.1, we revisit the contributions of the presented work, and in section 6.2, we propose potential extensions to be explored in the future.

6.1 Contributions Revisited

Reliability, the ability of a system to performance its required functions for a specified period of time, is the most important requirement in many of mission-critical systems. However, building WSN applications with high reliability is notoriously difficult. This tasks becomes more difficult in mission critical systems. These systems are usually large and dense systems, consisting of many nodes deployed closely to cover fields of interests. Second, nodes in these systems usually possess limited CPUs, memories, and sensors all driven by small batteries. Also, they are equipped with less powerful radio components and share the limited bandwidth with neighboring nodes. Third, mission-critical systems are usually event-driven and generate high volume data and burst traffic. Forth, these systems, deployed indoor or fields, also suffer the uncertainty of impacts of unpredictable environment and human factors.

This dissertation addresses two fundamental reliability issues for WSN: how to provide reliable and efficient communication and how to guarantee and maintain expected application-level correctness over system lifetimes. Contribution of this dissertation lie in the combination of an automated design framework, novel reliability methodologies and components for networking and application levels that will support the design and implementation of more reliable and robust WSNs. Providing such systems is one of the next steps to seeing WSNs widely used in more mission-critical applications.

1. *Contributions in Multi-channel communication:* we are the first to investigate multi-channel realities through empirical experiments and asset its impact on existing multi-channel protocols. We identifies three key practical issues for WSN multi-channel designs, which are insufficient available channels, overhead of channel switching and coordination, and sensitivity of time synchronization errors.

   We propose a novel idea to use multiple channels in WSNs. This new multi-channel methodology eliminates the need of time synchronization and the complexity and overhead for channel switching and coordination, and separate channel assignment from MAC-layer design and provides simplicity and flexibility to use any existing MAC protocols, while still enjoying the benefit from parallel communication of multiple channels.

   Next, we propose network partition and channel assignment algorithms to further optimize network metrics. We first study an optimization problem which aims to find the best assignment that minimizes interference within sub-trees. We prove it a NP-hard problem and propose a greedy algorithm that yield near-optimal results. Furthermore, we take the link diversity into consideration, and propose an efficient algorithm to minimize interference as well as to meet end-to-end reliability requirements in networks with low-quality links.

   We implement a novel multi-channel protocol, TMCP in both simulation and real MicaZ testbed, which combine aforementioned ideas and algorithms. Our development and evaluation demonstrates that 1) TMCP efficiently improve network performance over other multi-channel protocols with less interferences, much higher packet delivery rates, and higher network throughput; 2) TMCP works well with high network density, small numbers of available channels and poor and diverse link qualities. As a result, TMCP is a efficient and practical multi-channel solution to provide reliable communication for mission-critical solutions.

   At last, we extend our multi-channel research to consider how to dynamically assign channels according to live traffic patterns at run-time. We propose a simple and efficient traffic-aware channel assignment scheme based on the assumption that where perfect information about current and future traffic is available. The new scheme exploit this information to minimize interference occurring with real traffic. We compare this scheme with two typical static channel assignment schemes by simulation, and results show that being traffic-aware can substantially improve the performance of channel assignment. This baseline analysis helps establish the potential benefits of traffic-aware channel assignment algorithms.

Our work provides important and practical guidelines for researcher to develop more sophisticated multi-frequency parallel communication for sensor networks in the future, and also contributes efficient and reliable solutions that can be used for many applications.

2. *Contributions in Packet Collision Recovery:* We are the first to conduct an empirical study on bit error patterns of collided packets in WSNs. Our study reveals LS-collision, which can be exploited to achieve efficient collision recovery.

   We provide a thorough analysis based on the model derived from scenarios where N senders compete for transmission using CSMA. The analysis demonstrates that by exploiting LS-collisions, we can achieve a higher probability of successful transmission, as well as a much higher transmission efficiency.

   We then presents ACR, an Active Collision Recovery protocol that mitigates the negative impact of collisions by efficiently recovering collided packets. Unlike other existing recovery schemes, ACR is active in that it actively transforms potential collisions into LS-collisions to maximize the recovery probability, rather than passively waiting for collided packets to be recovered. ACR then uses a block based FEC scheme to efficiently recover corrupted packets due to LS-collisions. To identify erroneous blocks, ACR employs a novel RSSI-based error detection method, which does not introduce extra checksum overhead. In case of recovery failures, ACR also has a backup ARQ scheme. Unlike ZigZag [36] or PPR [46], ACR does not require customized hardware. ACR can be easily integrated into existing CSMA protocols with off-theshelf sensor devices.

We implement and evaluate the ACR prototype on a Tmote testbed. Results demonstrate that ACR achieves 25% higher transmission efficiency than existing recovery schemes.

The work we present in this chapter opens a new way of recovering packet collision and hence improving communication throughout and energy efficiency in wireless sensor networks.

3. *Contributions in Run Time Assurance:* Continuous and reliable operation of WSNs is notoriously difficult to guarantee due to hardware degradation and environmental changes. we propose Run Time Assurance (RTA), a new design principle that requires that systems can be periodically validated to demonstrate their run time operability and the integrity of their application semantics. The essence of run time assurance (RTA) lies in the systems capability to identify failures at the application-level on a periodical basis or by request.

   We propose and demonstrate a methodology for *run-time assurance* (RTA), in which we validate at run time that a WSN will function correctly, irrespective of any changes to operating conditions since it was originally designed and deployed. The basic approach is to use program analysis and compiler techniques to facilitate automated testing of a WSN at run time.

   We have implemented a framework for designing and automatically testing WSN applications using the RTA methodology. This framework adapts a high-level Sensor Network Event Description Language (SNEDL) [50] that permits designers to specify the system logic as well as the RTA requirements. The language addresses application semantics and monitoring needs for various mechanisms. Importantly, it also permits automatic code generation. Our system compiles the SNEDL model down to TinyOS [61] code that runs on the Telos nodes [24], as well as tests the defined mappings between sensor input values and system outputs.

   We evaluate our implementation by designing a fire detection system and executing it on a network of 21 Telos nodes. We artificially introduce failures into the system, including node failures and location errors, and compare the performance of RTA to that of an existing health monitoring solution [86]. Our results indicate that RTA misses 75% fewer system failures and also produces 70% fewer maintenance dispatches than health monitoring.

We believe that the broad impact of this work can be extensive since there is a proliferation of embedded systems being deployed or contemplated for critical applications such as fire fighting, pollution control, disaster response, tracking, military surveillance, and medical assistance. Providing systems with the capability to certify that they are operational is one of the next steps to seeing such technology widely used. Without effective run time assurances, systems will be unsafe or just not deployed in many situations.

Our work in this dissertation paves the way for networking tiny and resource limited sensor nodes into high-confidence systems for mission-critical applications. Models and protocols we develop in the networking layer lead to more efficient and reliable wireless communication so that high-volume and burst data streams can be efficiently transferred across networks timely. New design principle, methodologies and highly automated framework we propose in runtime assurance help designer and users verify an application's integrity at runtime, and build reliable and trustful WSN systems. In general, by developing novel protocols and methodologies in networking and application levels, an overall reliability enhancement of wireless sensor systems is achieved, which sets up a solid basis for developing wireless sensor networks technology to more mission-critical applications. Four major conference publications have come from this dissertation.

6.2 Potential Extensions

In this section we suggest potential extensions for future work, for the individual methodology we have proposed.

- *Potential Extensions for Multi-channel Communication:* Current sensor network nodes are equipped with one single radio transceiver and can only use one single channel at one time. We believe that future sensor nodes can possess multiple radio components, so that they can send and receive with different channels at the same time. With such hardware, new networking protocols are needed to exploit simultaneous multi-channel transmission to further increase the bandwidth and improve network efficiency.

In our current work, we assume that there are no discrepancies of link quality among different channels. However, it may not be always true. We have observed that channels show different link characteristics in some indoor experiments. It will be interesting to further study this phenonom and redesign channel assignment or routing algorithm with considering this realistic factor.

In addition, we also plan to extend TMCP to the case when the multiple channels have partially overlapping frequency bandwidths, instead of having well separated frequency bandwidths. In this case, a node that runs on an overlapping frequency bandwidth has the ability of direct communication with nodes that work on multiple frequencies.

- *Potential Extensions for Packet Collision Recovery:* Currently, ACR is only used to recover the collision packets between sensor nodes. In reality, we found interference and collision also occur among sensor nodes, invidiual wireless devices and electric appliances like microwaves co-exist in a future home or office, because they all share 2.4GHz unlicensed ISM wireless bandwidth. We believe that we can extend ACR to recover corrupted WSN packets in such cases.

- *Potential Extensions for Run Time Assurance:* Our RTA framework effectively helped users build high confidence WSN application by automating code generation and test execution, which has been demonstrated with example firefighting applicaitons. We plan to apply our framework into more complicated applications, and extend the SNEDL language and automated code generator to handle more sophisticted application semantics and logics.

  We also suggest that sometimes it is possible to specify that if an application level function can not be demonstrated, then the system should perform a set of lower level (system level) assurances to help find the cause of the problem. To find complicated root causes it needs to collect the monitored data into a data warehouse. When executing assurances we can then identify when the assurances show that the system is operational and when not. Based on this information we can construct models of the system and its operation. Then, via standard data mining algorithms we can search for conditions that are common for successful tests and those for unsuccessful tests.

Bibliography

[1] A. Adya, P. Bahl, J. Padhye, A.Wolman, and L. Zhou. A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks. In *IEEE Broadnets 2004*, 2004.

[2] G. S. Ahn, A. Campbell, A. Veres, and L. H. Sun. Supporting Service Differentiation for Real-time and Best Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN). In *IEEE Transactions on Mobile Computing*, 2002.

[3] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo. Funneling-MAC: A Localized, Sink-Oriented MAC for Boosting Fidelity in Sensor Networks. In *ACM SenSys*, 2006.

[4] J. Ahn, S. Hong, and J. Heidemann. An Adaptive FEC Code Control Algorithm for Mobile Wireless Sensor Networks. *Journal of Communications and Networks*, 2005.

[5] Y. J. Al-Raisi and D. J. Parish. Approximate wireless sensor network health monitoring. In *IWCMC*, August 2007.

[6] Ambient-systems unode product sheet. http://www.ambient-systems.net.

[7] G. Mainland B. Chen, G. Peterson and M. Welsh. LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics. In *DCOSS 2008*, June 2008.

[8] P. Bahl, R. Chancre, and J. Dungeon. SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks. In *ACM MobiCom 2004*, September 2004.

[9] M. Bahrepour, N. Meratnia, and P. J. M. Havinga. Automatic Fire Detection: A Survey from Wireless Sensor Network Perspective. In *Technical Report TR-CTIT-08-73, Centre for Telematics and Information Technology, University of Twente, Enschede*, 2008.

[10] S. Bapat, V. Kulathumani, and A. Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. In *IEEE ICNP*, 2005.

[11] R. Battiti, A. A. Bertossi, and M. Brunato. Cellular Channel Assignment: A New Localized and Distributed Strategy. In *Mobile Networks and Applications 2001*, 2001.

[12] S. Beirne, B. Corcoran, K. Lau, and D. Diamond. Chemical event tracking using a low-cost wireless chemical sensing network. In *IEEE Sensors 2008*, October 2008.

[13] Berkeley FireBug. http://firebug.sourceforge.net/.

[14] L. Bernardo, R. Oliveira, R. Tiago, and P. Pinto. A Fire Monitoring Application for Scattered Wireless Sensor Networks: A peer-to-peer cross-layering approach. In *WINSYS*, July 2007.

[15] M. Buettner, G. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *ACM SenSys*, 2006.

[16] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *ACM SenSys 2006*, November 2006.

[17] M. Burkhart, P. V. Rickenbach, R. Wattenhofer, and A. Zollinger. Does topology control reduce interference. In *ACM MobiCom*, 2004.

[18] A. Campbell C. Wan, S. Eisenman. CODA: Congestion Detection and Avoidance in Sensor Networks. In *SenSys*, November 2003.

[19] M. Caccaco, L. Y. Zhang, L. Sha, and G. Buttazzo. An Implicit Prioritized Access Protocol for Wireless Sensor Networks. In *IEEE RTSS 2002*, December 2002.

[20] CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://www.chipcon.com.

[21] X. Chen, P. Han, Q. He, S. Tu, and Z. Chen. A Multi-Channel MAC Protocol for Wireless Sensor Networks. 2006.

[22] K. Chintalapudi, J. Paek, S. Rangwala N. Kothari, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring Civil Structures with a Wireless Sensor Network . In *IEEE Internet Computing*, March/April 2006.

[23] P. Crescenzi, G. Gambosi, and P. Penna. On-line algorithms for the channel assignment problem in cellular networks. In *Discrete Applied Mathematics 2004*, 2004.

[24] CrossBow. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[25] Crossbow micaz - wireless measurement system. http://www.xbow.com/Products/Product_pdf_ files/Wireless_pdf/MICAz Datasheet.pdf.

[26] T. Dam and K. Langendoen. An Adaptive Evergy-Sufficient MAC Protocol for Wireless Sensor Networks. In *ACM SenSys 2003*, November 2003.

[27] J. Deng and Z. Haas. Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks. In *IEEE ICUPC*, October 1998.

[28] Technical Overview of Time Synchronized Mesh Protocol (TSMP). http://www.dustnetworks.com.

[29] M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, and P. Blum. Deployment Support Network - A Toolkit for the Development of WSNs. In *EWSN 2007*, January 2007.

[30] A. El-Hoiyi, J.-D. Decotignie, and J. Hernandez. Low Power MAC Protocols for Infrastructure Wireless Sensor Networks. In *The Fifth European Wireless Conference*, February 2004.

[31] K. Whitehouse et. al. Marionette: Using RPC for Interactive Development and Debugging of Wireless Embedded Networks. In *IPSN*, April 2006.

[32] F. Fitzek, D. Angelini, G. Mazzini, , and M. Zorzi. Design and Performance of an Enhanced IEEE 802.11 MAC Protocol for Multihop Coverage Extension. In *IEEE Wireless Communications*, 2003.

[33] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four bit wireless link estimation. In *HotNets*, 2007.

[34] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher. Datalink streaming in wireless sensor networks. In *ACM SenSys*, 2006.

[35] C. Girault and R. Valk. *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[36] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In *ACM SIGCOMM*, 2008.

[37] A. Arora H. Zhang. GS3: Scalable Self-Configuration and Self-Healing in Wireless Sensor Networks. In *Computer Networks (Elsevier)*, November 2003.

[38] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, T. Yan, J. Hui, and B. Krogh. Energy Efficient Surveillance Systems Using Wireless Sensor Networks. In *Mobisys*, June 2004.

[39] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *ACM MobiSys 2004*, 2004.

[40] D. Herbert, V. Sundaram, Y. Lu, S. Bagchi, and Z. Li. Adaptive Correctness Monitoring for Wireless Sensor Networks Using Hierarchical Distributed Run-Time Invariant Checking. In *TAAS*, Sept. 2007.

[41] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In *The Nineth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, November 2000.

[42] IBM Autonomic Computing, 2008. http://www.research.ibm.com/ autonomic/.

[43] IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, 1999. ANSI/IEEE Std. 802.11, 1999.

[44] IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003. http://www.ieee802.org/15/pub/TG4.html.

[45] N. Jain and S. R. Das. A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks. In *IEEE IC3N*, October 2001.

[46] K. Jamieson and H. Balakrishnan. Ppr: Partial packet recovery for wireless networks. In *ACM SIGCOMM*, 2007.

[47] K. Jamieson, H. Balakrishnan, and Y. C. Tay. Sift: A mac protocol for event-driven wireless sensor networks. Technical report, MIT Laboratory for Computer Science, 2003.

[48] J. Janssen, D. Krizanc, L. Narayanan, and S. Shende. Distributed Online Frequency Assignment in Cellular Networks. In *Journal of Algorithms 2000*, 2000.

[49] X. Jiang, J. Taneja, J. Ortiz, A. Tavakoli, P. Dutta, J. Jeong, D. Culler, P. Levis, , and S. Shenker. An Architecture for Energy Management in Wireless Sensor Networks. In *SIGBED*, April 2007.

[50] B. Jiao, S. Son, and J. Stankovic. GEM: Generic event service middleware for wireless sensor networks. *INSS*, E83-A(11), June 2005.

[51] B. Karp. *Geographic Routing for Wireless Networks*. PhD thesis, Harvard University, Cambridge, MA, 2000.

[52] M. Khan, H. Le, H. Ahmadi, T. F. Abdelzaher, and J. Han. Dustminer: troubleshooting interactive complexity bugs in sensor networks. In *SenSys*, 2008.

[53] K. Klues, G. Hackmann, O. Chipara, and C. Lu. A Component Based Architecture for Power-Efficient Media Access Control in Wireless Sensor Networks. In *ACM SenSys 2007*, November 2007.

[54] A. Kolawa and D. Huizinga. *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press, 2007.

[55] N. Kurata, B. F. Spencer, M. Ruiz-Sandoval Jr., Y. Miyamoto, and Y. Sako. A Study on Building Risk Monitoring Using Wireless Sensor Network MICA-Mote. In *ISHMII*, November 2003.

[56] P. Kyasanur and N. H. Vaidya. Capacity of multi-channel wireless networks: impact of number of channels and interfaces. In *ACM MobiCom*, 2005.

[57] Z. Lai, S. Cheung, and W. Chan. Inter-context control-flow and data-flow test adequacy criteria for nesc applications. In *SIGSOFT*, 2008.

[58] J. Nogueira L.B. Ruiz and A. Loureiro. MANNA: A Management Architecture for Wireless Sensor Networks. In *IEEE Communications Magazine*, Feb. 2003.

[59] H. K. Le, D. Henriksson, and T. F. Abdelzaher. A Control Theory Approach to Throughput Optimization in Multi-Channel Collection Sensor Networks. In *ACM/IEEE IPSN'07*, April 2007.

[60] H. K. Le, D. Henriksson, and T. F. Abdelzaher. A control theory approach to throughput optimization in multi-channel collection sensor networks. In *ACM/IEEE IPSN*, 2007.

[61] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse1, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *Ambient Intelligence*, 2005.

[62] J. Li, Z. J. Haas, M. Sheng, , and Y. Chen. Performance Evaluation of Modified IEEE 802.11 MAC for Multi-Channel Multi-Hop Ad Hoc Network. In *IEEE AINA 2003*, 2003.

[63] M. Li and Y. Liu. Underground structure monitoring with wireless sensor networks. In *ACM/IEEE IPSN*, 2007.

[64] S. Lin, G. Zhou, K. Whitehouse, Y. Wu, J. A. Stankovic, and T. He. Toward Stable Network Performance in Wireless Sensor Networks. In *IEEE RTSS*, 2009.

[65] H. Liu, L. Selavo, , and J. Stankovic. SeeDTV: Deployment-Time Validation for Wireless Sensor Networks. In *EmNetS 2007*, June 2007.

[66] L. Luo, T. He, G. Zhou, L. Gu, T. Abdelzaher, and J. A. Stankovic. Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog. In *Infocom 2006*, April 2006.

[67] M. Franklin. Declarative interfaces to sensor networks. Presentation at NSF Sensor Workshop, 2004.

[68] K. Gupta M. Khan, T. Abdelzaher. Towards Diagnostic Simulation in Sensor Networks. In *DCOSS*, June 2008.

[69] A. Vitaletti M. Ringwald, K. Romer. Passive Inspection of Sensor Networks. In *DCOSS 2007*, June 2007.

[70] A. Vitaletti M. Ringwald, K. Romer. SNTS: Sensor Network Troubleshooting Suite. In *DCOSS*, June 2007.

[71] M. Merabti M. Yu, H. Mokhtar. Fault Management in Wireless Sensor Networks. In *IEEE Wireless Communications*, Dec. 2007.

[72] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh. Partially Overlapped Channels Not Considered Harmful. In *ACM SigMetrics*, 2006.

[73] A. Nasipuri and S. R. Das. Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks. In *IEEE Vehicular Technology Conference*, September 2000.

[74] A. Nasipuri, J. Zhuang, and S. R. Das. A Multichannel CSMA MAC Protocol for Multihop Wireless Networks. In *IEEE WCNC*, September 1999.

[75] N. Nguyen and M. L. Soffa. Program representations for testing wireless sensor network applications. In *DOSTA*, 2007.

[76] Nordic semi-conductors, nrf905 multiband transceiver. http://www.nordicsemi.com.

[77] J. Paek, O. G. K. Jang, D. Nishimura, R. Govindan, J. Caffrey, M. Wahbeh, and S. Masri. A Programmable Wireless Sensing System for Structural Monitoring. In *4WCSCM*, July 2006.

[78] L. Paradis and Q. Han. A Survey of Fault Management in Wireless Sensor Networks. In *JNSM*, June 2007.

[79] M. Petrova, L. Wu, P. Mahonen, and J. Riihijarvi. Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks. In *IEEE ICN*, 2007.

[80] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Median Access for Wireless Sensor Networks. In *ACM SenSys*, 2004.

[81] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. In *ACM SenSys 2003*, November 2003.

[82] N. Ramanathan, K. Chang, L. Girod, R. Kapur, E. Kohler, , and D. Estrin. Sympathy for the sensor network debugger. In *SenSys*, November 2005.

[83] A. Raniwala and T. Chiueh. Architecture and Algorithm for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network. In *IEEE INFOCOM 2005*, March 2005.

[84] J. Regehr. Random testing of interrupt-driven software. In *EMSOFT*, 2005.

[85] Reliability Definition. http://en.wikipedia.org/wiki/Reliability.

[86] S. Rost and H. Balakrishnan. Memento: A Health Monitoring System for Wireless Sensor Networks. In *SECON 2006*, Sept. 2006.

[87] L. Ruiz, I. Siqueira, L. Oliveira, H. Wong, J. Nogueira, and A. Loureiro. Fault Management in Event-Driven Wireless Sensor Networks. In *MSWiM*, October 2004.

[88] D. Sexton, M. Mahony, M. Lapinski, and J. Werb. Radio channel quality in industrial wireless sensor networks. In *IEEE SICON*, 2005.

[89] D. Shin, S. Y. Na, J. Y. Kim, and S. Baek. Fish Robots for Water Pollution Monitoring Using Ubiquitous Sensor Networks with Sonar Localization. In *ICCIT*, Nov. 2007.

[90] H. W. So, G. Nguyen, and J. Walrand. Practical synchronization techniques for multi-channel MAC. In *ACM MobiCom*, 2006.

[91] J. So and N. Vaidya. Multi-Channel MAC for Ad-Hoc Networks: Handling Multi-Channel Hidden Terminal Using A Single Transceiver. In *ACM MobiHoc 2004*, May 2004.

[92] T. Sookoor, T. Hnat, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrodebugging: Global views of distributed program execution. In *SenSys*, 2009.

[93] P. Ramanathan T. Clouqueur, K. K. Saluja. Fault Tolerance in Collaborative Sensor Networks for Target Detection. In *IEEE Transactions on Computers*, March 2004.

[94] Z. Tang and J.J. Garcia-Luna-Aceves. Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks. In *IEEE INFOCOM 1999*, March 1999.

[95] Y. C. Tay, K. Jamieson, and H. Balakrishnan. Collision-Minimizing CSMA and its Applications to Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 2004.

[96] Telosb mote platform. http://www.xbow.com/Products/Productpdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[97] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *EWSN*, Feb. 2005.

[98] A. Tzamaloukas and J. J. Garcia-Luna-Aceves. A Receiver-Initiated Collision-Avoidance Protocol for Multi-Channel Networks. In *IEEE InfoCom*, 2001.

[99] R. Vedantham, S. Kakumanu, S. Lakshmanan, and R. Sivakumar. Component based channel assignment in single radio, multi-channel ad hoc networks. In *ACM MobiCom*, 2006.

[100] P. Wan and M. Lemmon. Distributed Flow Control using Embedded Sensor-Actuator Networks for the Reduction of Combined Sewer Overflow (CSO) Events. In *IEEE CDC*, December 2007.

[101] Y. Wang, J. P. Lynch, and K. H. Law. A Wireless Structural Health Monitoring System with Multithreaded Sensing Devices: Design and Validation. In *Structure and Infrastructure Engineering*, June 2007.

[102] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *ACM Sensys*, 2003.

[103] A. D. Wood, J. A. Stankovic, and G. Zhou. DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks. In *IEEE SECON 2007*, June 2007.

[104] S.-L. Wu, C.-Y. Liu, Y.-C. Tseng, and J.-P. Shen. A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks. In *I-SPAN*, 2000.

[105] Y. Wu, K. Kapitanova, J. Li, J. A. Stankovic, S. Son, and K. Whitehouse. Run Time Assurance of Application-Level Requirements in Wireless Sensor Networks. In *IEEE/ACM IPSN*, 2010.

[106] Y. Wu, J. A. Stankovic, T. He, and S. Lin. Realistic and Efficient Multi-channel Communication in Wireless Sensor Networks. In *IEEE INFOCOM*, 2008.

[107] Y. Wu, G. Zhou, and J. Stankovic. ACR: Active Collision Recovery in Dense Wireless Sensor Networks. In *INFOCOM*, 2010.

[108] W. Xu, W. Trappe, and Y. Zhang. Channel Surfing: Defending Wireless Sensor Networks from Interference. In *ACM/IEEE IPSN'07*, April 2007.

[109] D. Estrin Y. Zhao, R. Govindan. Residual Energy Scan for Monitoring Sensor Networks. In *WCNC*, March 2002.

[110] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance for Sensor Networks. In *ACM SenSys 2003*, November 2003.

[111] J. Yang, M.L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks. In *SenSys*, November 2007.

[112] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *IEEE INFOCOM*, 2002.

[113] W. Ye, F. Silva, and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *ACM SenSys 2006*, November 2006.

[114] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing*, 2007.

[115] J. Zhang, G. Zhou, C. Huang, S. Son, and J. A. Stankovic. TMMAC: An Energy Efficient Multi-Channel MAC Protocol for Ad Hoc Networks. In *IEEE ICC*, 2007.

[116] J. Zhang, G. Zhou, S. Son, and J. A. Stankovic. Ears on the Ground: An Acoustic Streaming Service in Wireless Sensor Networks. In *IEEE/ACM IPSN Demo Abstract*, 2006.

[117] G. Zhou, T. He, J. A. Stankovic, and T. F. Abdelzaher. Radio Interference Detection in Wireless Sensor Networks. In *IEEE INFOCOM*, 2005.

[118] G. Zhou, C. Huang, T. Yan, T. He, and J. A. Stankovic. MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks. In *IEEE Infocom*, 2006.

[119] G. Zhou, J. A. Stankovic, and S. Son. Crowded Spectrum in Wireless Sensor Networks. In *IEEE EmNets*, 2006.

[120] G. Zhou, Y. Wu, T. Yan, T. He, C. Huang, J. A. Stankovic, and T. Abdelzaher. A multifrequency mac specially designed for wireless sensor network applications. *ACM Trans. Embed. Comput. Syst.*, 9(4):1–41, 2010.