January 2010

Towards Real-time Wireless Sensor Networks

Octav Chipara
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Chenyang Lu, Chair
Gruia-Catalin Roman
Roger D. Chamberlain
William D. Smart
Thomas C. Bailey
John Stankovic

TOWARDS REAL-TIME WIRELESS SENSOR NETWORKS

by

Octav Chipara

A dissertation presented to the Graduate School of Arts and Sciences
of Washington University in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2010
Saint Louis, Missouri

ABSTRACT OF THE THESIS

Towards Real-time Wireless Sensor Networks

by

Octav Chipara

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2010

Research Advisor: Professors Chenyang Lu and Gruia-Catalin Roman

Wireless sensor networks are poised to change the way computer systems interact with the physical world. We plan on entrusting sensor systems to collect medical data from patients, monitor the safety of our infrastructure, and control manufacturing processes in our factories. To date, the focus of the sensor network community has been on developing best-effort services. This approach is insufficient for many applications since it does not enable developers to determine if a system's requirements in terms of communication latency, bandwidth utilization, reliability, or energy consumption are met. The focus of this thesis is to develop real-time network support for such critical applications.

The first part of the thesis focuses on developing a power management solution for the radio subsystem which addresses both the problem of idle-listening and power control. In contrast to traditional power management solutions which focus solely on reducing energy consumption, the distinguishing feature of our approach is that it achieves both energy efficiency and real-time communication. A solution to the

idle-listening problem is proposed in *Energy Efficient Sleep Scheduling based on Application Semantics* (ESSAT). The novelty of ESSAT lies in that it takes advantage of the common features of data collection applications to determine when to turn on and off a node's radio without affecting real-time performance. A solution to the power control problem is proposed in *Real-time Power Aware-Routing* (RPAR). RPAR tunes the transmission power for each packet based on its deadline such that energy is saved without missing packet deadlines.

The main theoretical contribution of this thesis is the development of novel transmission scheduling techniques optimized for data collection applications. This work bridges the gap between wireless sensor networks and real-time scheduling theory, which have traditionally been applied to processor scheduling. The proposed approach has significant advantages over existing design methodologies: (1) it provides predictable performance allowing for the performance of a system to be estimated upon its deployment, (2) it is possible to detect and handle overload conditions through simple rate control mechanisms, and (3) it easily accommodates workload changes. I developed this framework under a realistic interference model by coordinating the activities at the MAC, link, and routing layers.

The last component of this thesis focuses on the development of a real-time patient monitoring system for general hospital units. The system is designed to facilitate the detection of clinical deterioration, which is a key factor in saving lives and reducing healthcare costs. Since patients in general hospital wards are often ambulatory, a key challenge is to achieve high reliability even in the presence of mobility. To support patient mobility, I developed the Dynamic Relay Association Protocol – a simple and effective mechanism for dynamically discovering the right relays for forwarding patient data – and a Radio Mapping Tool – a practical tool for ensuring network coverage in

802.15.4 networks. We show that it is feasible to use low-power and low-cost wireless sensor networks for clinical monitoring through an in-depth clinical study. The study was performed in a step-down cardiac care unit at Barnes-Jewish Hospital. This is the first long-term study of such a patient monitoring system.

Acknowledgments

I would like to thank my advisers, Professor Chenyang Lu and Professor Catalin Roman, who have guided my steps through this journey. I am grateful for their sage advice and for allowing me to explore the many topics that sparked my imagination. I would also want to thank Professor John Stankovic whose kind words and insights have been greatly appreciated. Finally, I would like to thank Professor Tom Bailey and the nursing staff at Barnes Jewish Hospitals – without their help, my work on patient monitoring systems, would not have been possible.

I am grateful to have had the opportunity to collaborate with some of the most intelligent people I know: Greg Hackmann, Guoliang Xing, Sangeeta Bhattacharya, and Liang Fok. I am proud to call them not only co-workers but also friends. Special thanks go to Haraldur Thorvaldsson and Paul Gross who have listened to my trite diatribe through the tougher times. Their moral support was invaluable.

I would like to thank my parents foremost. Without their love and support, none of this would have been possible. They always inspired me to strive for better. Last but not least, I want to thank my fiance, Serena Ahrens. She has been my favorite distraction.

Octav Chipara

Washington University in Saint Louis
May 2010

Dedicated to my parents

Contents

Abstract . . . . . **ii**

Acknowledgments . . . . . **v**

List of Figures . . . . . **xi**

1 Introduction **1**
1.1 Contention-based Real-time Communication 4
1.2 Predictable Real-time Data Collection 5
1.3 Reliable Wireless Clinical Monitoring 6
1.4 Research Contributions 7

2 Efficient Power Management based on Application Timing Semantics **9**
2.1 Related Work 11
2.2 Workload Model 12
2.3 Protocols Design 13
2.3.1 Safe Sleep (SS) 14
2.3.2 Traffic shapers 17
2.3.3 Protocol Maintenance 23
2.4 Experiments 25
2.4.1 Energy Efficiency 26
2.4.2 Query Performance 28
2.4.3 Impact of Break-Even-Time of Radio. 30
2.5 Summary 31

3 Real-time Power-Aware Routing **33**
3.1 Impact of Transmission Power on Delay 34
3.1.1 Empirical Study on XSM2 Motes 34
3.1.2 Tradeoff between Delay and Capacity 36
3.2 Problem Formulation 37
3.3 Design of RPAR 38
3.3.1 Dynamic Velocity Assignment Policy 38
3.3.2 Forwarding Policy 39
3.3.3 Delay Estimator 40
3.3.4 Neighborhood Manager 41

3.4 Experimental Evaluation . . . 46
3.4.1 Performance of Forwarding Policy . . . 48
3.4.2 Performance with Neighborhood Management . . . 48
3.4.3 Impact of Workload . . . 51
3.5 Discussion . . . 51
3.5.1 Handling Congestion . . . 53
3.5.2 Handling Holes . . . 53
3.5.3 Integration with Power Management . . . 54
3.6 Related Work . . . 54
3.7 Summary . . . 56

4 Dynamic Conflict-free Query Scheduling . . . 57
4.1 System Models . . . 58
4.1.1 Query Model . . . 58
4.1.2 Network Model . . . 59
4.2 Protocol Design . . . 62
4.2.1 The Centralized Planner . . . 64
4.2.2 Plan Sharing . . . 68
4.2.3 The Scheduler . . . 70
4.2.4 The Multiple Query Class Scheduler . . . 76
4.2.5 Distributed Planner . . . 78
4.3 Handling Dynamics . . . 80
4.3.1 Dynamic Workload . . . 80
4.3.2 Preventing Overload . . . 80
4.3.3 Robustness Against Network Changes . . . 81
4.3.4 Robustness Against Variations in Link Quality . . . 82
4.3.5 Supporting Other Traffic . . . 83
4.3.6 Time Synchronization . . . 83
4.4 Performance Evaluation . . . 83
4.4.1 Single Query Class . . . 86
4.4.2 Multiple Query Classes . . . 93
4.5 Related Work . . . 97
4.6 Summary . . . 98

5 Real-Time Query Scheduling . . . 99
5.1 Related Work . . . 100
5.2 Real-time Query Scheduling . . . 102
5.2.1 Constructing plans . . . 103
5.2.2 Overview of Scheduling Algorithms . . . 105
5.2.3 Nonpreemptive Query Scheduling . . . 106
5.2.4 Preemptive Query Scheduling . . . 108
5.2.5 Analysis of NQS and PQS . . . 112
5.2.6 Slack Stealing Query Scheduling . . . 115

5.3 Handling Multiple Plans . . . 119
5.3.1 Multi-class NQS . . . 119
5.3.2 Multi-class PQS . . . 121
5.3.3 SQS Multi-class Scheduler . . . 124
5.4 Handling Packet Loss and Topology Changes . . . 126
5.5 Simulations . . . 126
5.5.1 Comparison with Baselines . . . 129
5.5.2 Comparison of RTQS Algorithms . . . 132
5.6 Summary . . . 134

6 Reliability Issues in a Wireless Clinical Monitoring System: A clinical trial . . . 136
6.1 Introduction . . . 136
6.2 Related Work . . . 138
6.3 System . . . 139
6.3.1 System Architecture . . . 140
6.3.2 Hardware . . . 141
6.3.3 Software Components . . . 142
6.4 Clinical Study . . . 146
6.4.1 Methods . . . 146
6.4.2 Reliability . . . 150
6.4.3 Benefits of Disconnection Alarms . . . 156
6.4.4 Detecting Clinical Deterioration . . . 158
6.5 Discussions . . . 159
6.6 Summary . . . 161

7 Practical Modeling and Prediction of Radio Coverage in Indoor Sensor Networks . . . 166
7.1 Introduction . . . 166
7.2 Related Work . . . 169
7.3 Radio Propagation Models . . . 170
7.4 Automatic Wall Classification . . . 174
7.5 Empirical Model Comparison . . . 176
7.5.1 Experimental Setup . . . 176
7.5.2 Effect of Walls . . . 178
7.5.3 Automatic Wall Classification . . . 179
7.5.4 Boosting . . . 181
7.5.5 Impact of Sectorization . . . 181
7.6 Radio Mapping Tool . . . 184
7.7 Empirical Evaluation of RMT . . . 187
7.7.1 Representative Example . . . 187
7.7.2 Detailed Empirical Results . . . 189
7.8 Summary . . . 190

8 Conclusions and Future Work **192**
8.1 Future Work 193

References **195**

List of Figures

2.1 Safe Sleep Algorithm 15
2.2 Impact of query deadline on duty cycle and query latency of STS-SS. 20
2.3 Overhead of DTS-SS in bytes per packet 22
2.4 Average duty cycle for three queries classes when varying base rate. . 27
2.5 Average duty cycle for three query classes when varying number of queries per class. 28
2.6 Distribution of duty cycles at different ranks. 29
2.7 Query latency for three query classes when varying base rate. 29
2.8 Query latency for three query classes when varying the number of queries per class. 30
2.9 Histogram of sleep intervals. Each point in the graph represents the number of sleep intervals whose length falls in the range $[x - 25, x]ms$. 31
2.10 Impact of distribution of sleep interval on STS-SS. Three queries are run at a base rate of $5Hz$. 31

3.1 Impact of transmission power and one-hop distance on delivery velocity. 35
3.2 Performance of considered protocols when deadline is varied. The neighborhood table is prefilled. 46
3.3 Performance of considered protocols when deadline is varied (with neighborhood management). 50
3.4 Performance of considered protocols when the workload is varied (with neighborhood management). 52

4.1 Conflicts in an IC graph 60
4.2 DCQS uses cross layer information in making scheduling decisions. It has two key components: a planner and a scheduler. 62
4.3 IC graph: The solid lines denote communication edges and form the routing tree. The dotted lines are interference edges. The edges without arrows are bi-directional. The shown numbers are the steps in which each node transmits under a plan for an instance with a workload demand of one slot per node. 65
4.4 Constructed plan for IC graph in Figure 4.3 when each node has a workload demand of one slot. The first and last column are the step indices in the reverse and actual plans, respectively. The top row indicates the intended receivers. The entries in the other rows indicate the senders in each step of the plan. 65

4.5 The centralized planner. 68
4.6 The conflict table captures the transmission conflicts between pairs of steps from the plan shown in Figure 4.4. The presence of a conflict is represented by the red rectangle. No rectangle indicates that the pair of steps may be executed concurrently. Based on the conflict table you can dynamically construct schedules either by brute force or through the DCQS approach. 72
4.7 Validation of capacity bound for single query. 84
4.8 Performance comparison when executing four queries and their base rate is varied. 88
4.9 Performance comparison for topologies of different sizes. 90
4.10 Communication costs for different size networks. 92
4.11 Impact of virtual transmissions on DCQS. 92
4.12 Validation the capacity bound for multiple query classes. 93
4.13 Performance comparison when three queries belonging to different query classes. 95

5.1 NQS pseudocode 108
5.2 PQS pseudocode 109
5.3 Scheduling with different prioritization policies. Workload: P_{hi}=30, D_{hi} =20,P_{med}=65, D_{med} =28, P_{lo}=93, D_{lo} =93. 110
5.4 Interference of I_h on I_l under PQS. 115
5.5 Pseudocode of multi-class PQS schedule. 121
5.6 Interference of I_h on I_l under multiclass PQS. 124
5.7 Response time of baselines, PQS, and NQS 127
5.8 Data fidelity of baselines, PQS, and NQS 128
5.9 Response time of queries when workload is varied by changing rates. All queries belong to the same class. 130
5.10 Response time of queries when workload is varied by changing the deadline of Q_0. All queries belong to the same class. 131
5.11 Response time of queries when workload is varied by changing the deadline of Q_0. Experiment includes multiple query classes 132

6.1 Hardware used for wireless clinical monitoring system 140
6.2 Deployment at Barnes-Jewish Hospitals. The blue square denotes the base station. Red circles denote relay nodes. 147
6.3 Network and sensing reliability per patient 149
6.4 Distribution of service intervals and outages for network component . 151
6.5 Distribution of service intervals and outages for the sensor component 153
6.6 Impact of movement on sensing 154
6.7 Impact of oversampling on sensing reliability 162
6.8 Expected performance of a sensor disconnection alarm system 163
6.9 Combining oversampling and sensor disconnection alarm systems . . . 164

6.10 Pulse (red) and oxygenation (purple) measurements from patients which suffered clinical deterioration 165

7.1 Wall classification algorithm 175
7.2 Test buildings 177
7.3 Comparison of propagation models 178
7.4 Automatic wall classification 180
7.5 Sectorization Model 182
7.6 Impact of distance, wall attenuation, and antenna orientation 183
7.7 Radio Mapping Tool 185
7.8 Selecting the RSS threshold 186
7.9 Example predictions using Radio Mapping Tool 188
7.10 Coverage prediction accuracy 189

Chapter 1

Introduction

Wireless sensor networks (WSNs) are networks composed of inexpensive embedded devices capable of sensing, radio communication, and computation. Such networks are poised to change the way computer systems interact with the physical world. We plan on entrusting WSNs with collecting data from patients, monitoring the safety of our infrastructure (e.g., bridges, levee), and controlling manufacturing processes in our factories. To date, the focus of the sensor network community has been on developing best-effort applications. Arguably, this approach is insufficient for critical applications since it does not enable developers to determine if a system's requirements in terms of communication latency, bandwidth utilization, reliability, or energy consumption are met. The focus of this thesis is to develop real-time network support for such critical applications.

WSN applications differ from traditional wireless applications in the type of communication patterns they exhibit. Mobile ad hoc networks focus on peer-to-peer communication and 802.11 networks focus on one-hop communication. In contrast, in a WSN one (or a few) nodes are selected as base stations[1] and connectivity is provided to/from the remaining nodes to these base stations. As a result, two types communication patterns emerge: *data collection* – the collection of data from nodes to a base station — and *data dissemination* – the dissemination of data from a base station to all or a subset of nodes. This thesis focuses on supporting real-time data collection as collecting data from a physical environment is the primary task of a sensor network. Much of this work is motivated by the need to support real-time data

[1]The selection of base stations is often due to practical reasons such as access to wired network infrastructure, availability of power, or easy access.

collection in the context of Structural Health Monitoring (SHM) [88] and Patient Monitoring [26] applications.

Real-time communication in WSNs faces several key challenges:

Varied and dynamic requirements: WSN applications have varied communication requirements. For example, in a patient monitoring application, different types of physiological measurements require different rates: pulse and oxygenation may measured once per minute whereas EKG requires sampling rates as high as 200 Hz. Moreover, data from different patients is not equally important from an application perspective: it is more important to deliver data from critical patients than from patients in stable condition. Therefore, it is critical for a real-time communication protocol to provide effective *prioritization* between different traffic classes while meeting their deadlines.

Interplay between requirements: The sensor networks community has been successful in developing solutions for improving the energy efficiency, reliability, and even supporting high data rates and meeting packet deadlines. However, these are point-solutions which focus on a single aspect of networking performance. As a result, often these solutions cannot be combined or, even worse, optimizing one dimension of networking performance may result in performance degradation across other dimensions. Therefore, it is necessary to move towards holistic solutions in which the interplay between requirements is explicitly captured.

Lossy and dynamic links: A serious impediment to developing real-time communication is the nature of low-power links. Empirical evidence shows that low-power wireless links have high error rates and that link quality may vary significantly. Common factors affecting link quality include interference among nodes, variations in background noise, and even people attenuating radio signals as they move through the environment. Most of these factors are outside the control of the system highlight the importance of including mechanisms for handling lossy and dynamic links. In contrast, interference may be handled through Media Access and Control (MAC) mechanisms. In this thesis, we consider the problem of providing real-time communication on top of both contention and time division MACs.

Stringent resource limitations: The promise of wireless sensor networks is to monitor the environment at unprecedented resolution. To make such a vision economically feasible sensor nodes must be inexpensive. As a result, typical sensor nodes have stringent hardware limitations (see Table 1.1). Of particular importance to use

	mica2	tmote	imote2
RAM	4 KB	10 KB	3200 KB
ROM	1 27KB	48 KB	3200 KB
CPU	8 bits/8 Mhz	16 bits/8 Mhz	32 bits/13-416Mhz
Radio	38.4 kbps	250 kbps	250 kbps

Table 1.1: Hardware capabilities of representative sensor nodes

are the limited energy budgets which allow the mote to operate only for a few days without power management and the limited radio bandwidth which becomes a limiting factor for high-data rate applications. Recognizing, these limitations is essential to developing practical solutions.

This thesis is organized in three broad parts. The first part – Chapter 2 and Chapter 3 – focuses on developing protocols for supporting real-time communication and energy efficiency on top of contention-based MACs. These protocols work by dynamically adapting their behavior in response to changes in the observed contention level. This is a particularly attractive option because of the flexibility of contention-based protocols to adapt to workload and topology changes. However, the random back-off scheme employed by contention-based MACs leads to highly variable communication latency and channel capacity. This makes a precise characterization of system performance difficult.

The second part of the thesis – Chapter 4 and Chapter 5 – focuses on the development of transmission scheduling techniques. Transmission scheduling is a classical problem in computer networking. In this thesis, we revisit this problem from a fresh perspective: develop transmission scheduling techniques optimized for data collection. A fundamental advantage of the propose scheduling techniques is that they enable the analysis of a system's performance when it is deployed.

The last part of the thesis – Chapter 6 and Chapter 7 – presents the development of a wireless patient monitoring system for collecting pulse and oxygenation from mobile in real-time. We focus on analyzing the reliability of the system through an in-depth clinical trial performed at Barnes-Jewish Hospital in St. Louis. An essential component of achieving robust data collection from mobile users is to ensure network coverage. The development of a practical Radio Mapping Tool for assessing network coverage is presented in Chapter 7.

The subsequent sections present the problems each part of the thesis considers. A high level overview of the devised solution focusing on its novel aspects is presented in the following. The section concludes by outlining the specific research contribution presented in this thesis.

1.1 Contention-based Real-time Communication

Energy is a scarce resource on battery operated nodes: the energy budge available on a sensor network provides enough power to operate a node only for a few days without power management. The radio subsystem contributes significantly to the energy budget of many applications. There are two key issues which must be addressed in radio power management. First, a significant energy may be consumed when the radio is idle listening i.e., when the radio is active but not packets are transmitted or received. During this time, the radio should be put in low-power states in order to conserve energy. Second, radios commonly transmit at various power levels and choosing the right power level may significantly reduce energy consumption. Through power control one may change link quality and degree of interference: by increasing the transmission power, poor quality links may be transformed in good quality links; unfortunately, this also results in additional interference. Reducing the transmission power has the opposite effect.

In *Efficient Sleep Scheduling based on Application Timing Semantics* (ESSAT) (see Chapter 2), I proposed a novel sleep scheduling technique. ESSAT is designed for data collection applications which sample sensors at known rates and use routing trees to deliver data to a base station. ESSAT works on a straight forward premise: if a node knows when it expects to receive packets from its children, then it could turn on its radio *just-in-time* to receive those packets; the difficulty is in determining the reception times efficiently. Since a data collection application samples sensors at known times, the key challenge is to maintain the predictability of packet receptions as packets are forwarded across multiple hops. ESSAT accomplishes this by introducing *light-weight traffic shapers* on each node, which, rather than forwarding a packet as soon as it is received, delay packet transmissions to effectively smooth out the jitter caused by CSMA/CA MAC protocols. ESSAT not only was more energy

efficient but it also reduced end-to-end communication delays by reducing the degree of interference between nodes at different depths in the routing tree. This is the result of tighter coordination between application, routing, and power management layers.

In *Real-time Power-Aware Routing* (RPAR) (see Chapter 3), I propose a power-aware routing protocol which adapts a packet's transmission power based on its deadline. RPAR highlights that power-control is an effective mechanism for meeting end-to-end deadlines under varying contention levels. Moreover, RPAR features a novel neighborhood discovery mechanism which enables real-time protocols to discover neighbors which enable the routing layer to meet the specified end-to-end deadlines.

1.2 Predictable Real-time Data Collection

The main theoretical contribution of this thesis is the development of a framework that provides predictable performance in terms of bandwidth, latency, and energy consumption for such applications. This is the first framework that *bridges the gap between wireless sensor networks and real-time scheduling theories* which have traditionally been applied to processor scheduling. The proposed framework has significant advantages over existing design methodologies: (1) it provides predictable performance allowing for the performance of a system to be estimated upon its deployment, (2) it is possible to detect and handle overload conditions through simple rate control mechanisms, and (3) it easily accommodates workload changes. I was able to develop the framework under a *realistic* interference model by coordinating the activities at the MAC, link, and routing layers. I developed novel transmission scheduling techniques which take advantage of temporal properties of the workload generated by data collection applications and integrated them with a static priority assignment scheme. My analysis revealed an interesting aspect of such a system: there is an inherent tradeoff between prioritization and throughput. I developed three scheduling algorithms which explore different trade-offs between prioritization and throughput. This framework promises to be a key building block for the next generation of cyber-physical systems that require predictable performance over large-scale wireless sensor networks.

1.3 Reliable Wireless Clinical Monitoring

The last component of this thesis focuses on the design, development, and evaluation of a wireless clinical monitoring system and its integration with an automatic early detection system. Early detection of clinical deterioration is a key factor in saving lives and reducing healthcare cost. While physicians are working on new automatic early detection systems (e.g., early detection of sepsis or heart/pulmonary attacks), the sensitivity and accuracy of such systems hinges upon having timely clinical data. This may not be a major problem for patients in Intensive Care Units since their vital signs are monitored by wired electronic monitoring systems. However, due to the significant cost of monitoring systems, the vital signs of patients in general hospital wards are collected manually at long intervals (e.g., hours) leading to prolonged delays in detecting clinical deterioration. Therefore, *a real-time and reliable clinical monitoring system for patients in general hospital wards* is critical for effective early detection of clinical deterioration.

I designed, implemented, and tested the communication stack for such a system. Our system has two types of nodes: relay nodes and patient nodes. A patient node integrates the TelosB embedded platform (with a microcontroller and 802.15.4 radio) with a pulse oximeter commercialized by Smiths Medical OEM. The relay nodes are deployed to ensure coverage in the clinic and form a multi-hop wireless mesh network. Since patients in general hospital wards are often ambulatory, a key challenge is to achieve high reliability even in the presence of mobility. Initial empirical studies have shown that the majority of packet losses occurred on the first-hop link between the patient node and the first-hop relays. Based on this insight I developed the Dynamic Relay Association Protocol (DRAP), a simple and effective mechanism for dynamically discovering the right relays for patient nodes without requiring any change to complex routing protocols.

The developed system was deployed used to collected data from 25 patients as part of clinical trial in a step-down cardiac care unit at Barnes-Jewish Hospital. The clinical trial was designed to better understand the reliability concerns of such a system in a real clinical environment. The clinical trial focused on answering five key questions:

- Does the system provide sufficient resolution for detecting clinical deterioration?

- What are the majors factors affecting system reliability?
- What is the distribution of networking and sensing failures?
- How much involvement from the nursing staff is necessary for operating the system?
- What are the power management issues in such a system?

The robust operation of the wireless clinical monitoring system depends on deploying relays to ensure wireless coverage. *Radio mapping* aims to predict network coverage based on a small number of link measurements. This problem is particularly challenging in complex indoor environments where walls significantly affect radio signal propagation. Nevertheless, we show that it is feasible to accurately predict coverage through a two-step process: a propagation model is used to predict signal strength at a recipient node, which is then mapped to a coverage prediction. Through in-depth empirical study, we show that complex models do *not* necessarily produce accurate estimates of signal strength: there is an important tradeoff between model accuracy and the number of parameters that must be estimated from limited training data. We find that the best performance is achieved by a family of models which classify walls based on attenuation into a small number of groups. A key feature of our approach is an algorithm for automatically classifying walls into classes with varying degrees of attenuation. Based on these insights, we build a novel *Radio Mapping Tool* (RMT) for predicting radio converge in indoor environments.

1.4 Research Contributions

Specifically, this thesis makes the following research contributions:

- **Design and analysis of an efficient sleep scheduling algorithm:** We devised a novel approach for sleep which takes advantage of the semantics of data collection application to decide when to turn on and off a node's radio. In contrast to other sleep scheduling algorithms, the proposed solution has minimal impact on the real-time performance of the system.

- **Design and analysis of a power-aware routing protocols:** We developed a novel approach for dynamically selecting the transmission power of a packet such that energy consumption is minimized while meeting packet deadlines. The proposed protocol may reduce energy consumption significantly when deadlines are lax.

- **Design and analysis of transmission scheduling techniques optimized for data collection:**. In contrast to traditional scheduling techniques which focus on scheduling packets, we developed novel transmission scheduling techniques optimized for data collection applications. These techniques are shown to significantly improve the performance of data collection applications in terms of latency, throughput, and energy efficiency.

- **Design and analysis of priority scheduling algorithms:** Our analysis of real-time data collection shows an inherent trade-off between prioritization and throughput. We designed three static priority scheduling algorithms with different trade-off points between prioritization and throughput. Schedulability analysis is provided for each algorithm. This analysis would enable a system developer to determine the system performance at deployment time.

- **Design, development, and implementation of a wireless clinical monitoring system:** We have developed and deployed a running wireless clinical monitoring system. The system has been successfully used a part of a clinical trial including over 25 patients.

- **A trial identifying the reliability issues in a clinical environment:** We performed an in-depth reliability study of our clinical monitoring system in a real clinical environment. The study shows that it is feasible to use low-power and low-cost wireless sensor networks for detecting clinical deterioration.

- **Designed and implemented a Radio Mapping Tool:** We have designed and implemented the first tool designed to ensure the coverage of low-power wireless sensor networks. Empirical studies show that it is feasible to predict network coverage based on a small number of measurements.

Chapter 2

Efficient Power Management based on Application Timing Semantics

Energy is the most critical resource in wireless sensor networks (WSNs) that must operate for years on limited power supplies. Recent studies have shown that significant energy savings can be achieved by dynamically managing node duty cycles. However, the design of power management protocols faces several key challenges. First, the network must maintain sufficient quality of service despite sleep schedules. In particular, many mission-critical applications operate under stringent timing constraints. For example, a surveillance application may require the network to report all suspicious events within a few seconds in order to ensure timely response to intrusions. Power management protocols designed for such applications must coordinate the sleep schedules of different nodes to minimize their impact on end-to-end communication delays. Second, hardware platforms in WSNs usually have limited bandwidth, memory, and processing capabilities. A practical power management protocol must be simple and introduce minimal overhead. Third, the workload in WSNs may change dramatically in response to events in the physical environment. For instance, while the workload in a fire monitoring system may be moderate during normal conditions, it may increase sharply after a wild fire is detected to support numerous fire fighting activities. Therefore, a power management protocol must dynamically adjust the duty cycles of nodes based on the current system workload. Furthermore, a power management protocol must be robust against node and link failures, which can occur frequently in WSNs [140].

We present *Efficient Sleep Scheduling based on Application Timing (ESSAT)*, a novel power management scheme that meets all the above challenges by aggressively exploiting *application timing semantics*. In contrast to general-purpose systems with random workloads, workloads in WSNs are often generated by applications with known timing semantics. A primary function of many WSN applications is to continuously gather data from the environment at user-specified periods. Moreover, in many distributed signal processing applications (e.g., target detection), multiple sensor nodes sample and exchange data at application-specific sampling frequencies for data fusion. Intuitively, a power management protocol can leverage such application-level timing semantics in order to optimize sleep schedules. However, the design of such protocols is complicated by several issues. The random backoff scheme in widely adopted CSMA/CA MAC protocols can cause variable communication delays due to channel contention. More importantly, the delay jitter can accumulate over multiple hops. As a result, even when data is generated periodically at sources, the workload often becomes highly *aperiodic* over multi-hop communication. In addition, the *aggregate* network workload of multiple periodic data flows may be *aperiodic* due to their different periods and starting times.

ESSAT deals with the above complexities with two efficient mechanisms: (1) *in-network traffic shapers* that actively control packet transmission to preserve predictable timing properties inside the network; and (2) a local sleep scheduler called *Safe Sleep* that wakes up nodes *just in time* to meet communication needs. ESSAT is optimized for WSNs. It saves significant energy while introducing minimal delay penalties. Our ESSAT protocols also address a number of important practical issues in WSNs. In contrast to many existing protocols (discussed in Section 2.1), our protocols do not maintain TDMA schedules or active communication backbones; therefore, they are highly efficient and suitable for resource constrained sensor platforms. Moreover, our protocols are robust in highly dynamic wireless sensor networks, i.e., they can adapt to varying workloads induced by multiple queries as well as node failures.

The remainder of the chapter is organized as follows. In Section 2.1, we highlight the contributions of ESSAT by contrasting it with related work. A workload model with application timing semantics is formulated in Section 2.2. Section 2.3 presents the design and analysis of Safe Sleep and two traffic shaper algorithms. Experimental

evidence regarding the effectiveness of the proposed protocols is discussed in Section 2.4. Concluding remarks appear in Section 2.5.

2.1 Related Work

ESSAT is based on two high-level design principles: (1) shaping the traffic inside a network to achieve predictable timing properties, and (2) exploiting application-level timing semantics. In the following, we discuss existing power management schemes and the extent to which they explore similar principles.

An approach adopted by several power management protocols is to maintain a connected communication backbone that is responsible for routing packets, while other nodes sleep most of the time [20, 136, 127]. Although the backbone provides good communication performance by maintaining sufficient network connectivity, this solution does not exploit the possibility of conserving energy on the backbone nodes even when they are not needed for communication. Keeping the backbone nodes continuously active may become unacceptable in WSNs especially when the workload is light. Power management schemes based on communication backbones do not exploit traffic shaping or workload characteristics.

The idea of using traffic shaping to facilitate sleep scheduling has been explored in power management schemes that operate at the MAC layer [102, 55, 138, 125, 124, 142]. Traffic shaping may be performed at different levels of granularity. TDMA MAC protocols [102, 55] perform fine-grained traffic shaping by allocating time slots to each node. A node only communicates in its slots, and can sleep in the others. However, maintaining fine-grained TDMA schedules for a multi-hop sensor network is challenging. Centralized scheduling algorithms cannot scale effectively in large networks while distributed scheduling algorithms can introduce significant synchronization overhead in order to maintain consistent schedules in multi-hop networks [102]. A simpler scheme for traffic shaping is based on coarse-grained sleep schedules [138, 59, 124, 142] in which each node follows a fixed periodic schedule that includes an active window and a sleeping window. However, none of the above traffic shaping schemes consider the workload characteristics when constructing sleep schedules.

Consequently, they may introduce significant delay penalties when their schedules interfere with the timing semantics of the application workload.

Several power management schemes exploit workload properties available at different layers to improve their energy efficiency. For example, T-MAC [125] and PSM [59] adapt a node's duty cycle in response to the network load observed at the MAC layer, while on-demand power management [141] uses routing information. However, neither solution is cognizant of the timing semantics at the application layer. As a result, they may introduce delay penalties or waste energy due to the lack of precise timing information of the workload.

TinyDB [89] allows a user to collect aggregated data from a sensor network through a routing tree. It evenly divides the period of a query into communication slots for nodes at different levels in the routing tree, and nodes can sleep in slots assigned to other levels. TinyDB does not address sleep scheduling for multiple queries with different timing properties. Moreover, the duty cycle of each node is fixed and does not adapt to the workload.

In contrast to the aforementioned approaches, ESSAT employs a novel power management approach based on light-weight traffic shaping and the timing semantics of WSN applications. The traffic shaping algorithms introduce minimal delay penalties and communication overhead. Furthermore, they can efficiently adapt to the dynamics in the network and aggregate workload of multiple queries. This unique combination of features makes ESSAT especially suitable for real-time applications on resource-constrained WSNs.

2.2 Workload Model

In this chapter we assume a general workload model in which each source produces data reports periodically for a query. This model fits many WSN applications that gather data from the environment at user specified rates. Such applications generally rely on existing WSN query services. We analyze the ESSAT protocols on the basis of the workloads produced by a generic query service.

A query is characterized by the following parameters: a set of sources that respond to registered queries, an aggregation function [89] for in-network aggregation; the period P at which data reports are generated by the sources; and the starting time of the query ϕ.

A query service usually works as follows: a user issues a query to a sensor network through a base station, which disseminates the query to all the sources. To facilitate data aggregation, the query service constructs a *routing tree* rooted at the base station as the query is disseminated. During the execution of the query, each leaf node generates a new data report every P seconds. The first data report is generated by the leaf nodes at time ϕ. Each non-leaf node waits to receive the data reports from its children, produces a new data report by aggregating its data with the children's data reports, and then sends it to its parent.[2]

Although in our model the sources produce data reports periodically for query we do not assume that the network workload remains periodic. In fact, as discussed in Section 2.4.3 the workload is aperiodic due to multi-hop delay jitter and multiple queries with different timing properties. Since this query model approximates several representative query services [61, 89], ESSAT can be easily integrated with existing query services to support various WSN applications. ESSAT can also be extended to support other communication patterns such as peer-to-peer communication or data dissemination.

2.3 Protocols Design

An ESSAT protocol has two components: a traffic shaper and a sleep scheduling algorithm called Safe Sleep. The traffic shaper controls the sending and receiving of data reports to construct workloads with predictable temporal properties. Based on the temporal properties of the workload, Safe Sleep determines when a node should be turned on or off.

[2]To deal with node failures, a node may timeout and send its data report to its parent before receiving the data reports from all its children (see Section 2.3.1).

The section starts by introducing Safe Sleep. We then consider the behavior of the WSNs in the absence of traffic shaping. Next, we introduce two traffic shapers that improve the energy efficiency. Finally, we analyze the behavior of the ESSAT protocols in the presence of packet drops and topology changes.

ESSAT is layered between the MAC protocol and the query service. ESSAT does not require special scheduling support at the MAC layer. For example, it can work with 802.11b and the CSMA/CA protocol of TinyOS [129].

2.3.1 Safe Sleep (SS)

Safe Sleep (SS) is a local sleep scheduling algorithm that turns the radio on and off. From the point of view of SS, a node may be in one of two states: *free* or *busy*. A node is *busy* when it expects to receive or send a data report. Otherwise, a node is *free*. SS determines a node's state based on the timing properties of its workload and schedules periods of sleep and activity accordingly.

Timing properties of queries. The timing properties of a node's query workload are shared by SS and the traffic shapers. A node characterizes each query as follows. Since the nodes are organized in a tree topology, a node expects a data report from each child in each query period. Let $r(q, k, c)$ be the *expected reception time* of the k^{th} data report for query q from child c. The *expected send time* of the k^{th} data report for query q, $s(q, k)$, is the time when the data report is scheduled to be submitted to the MAC layer for transmission. SS keeps track of all queries routed through a node. For each query q, the node stores the time it expects the next data report from each child in $q.r_{next}(c)$ and the time it expects to send the next aggregated data report to its parent in $q.s_{next}$. It is the responsibility of the traffic shaper to control the times when the next data reports are to be received or sent. This information is provided to SS in an incremental manner by the traffic shaper as follows. Upon receiving a data report for query q from child c, the traffic shaping protocol computes $r(q, c, k + 1)$ while upon completing the sending of a data report the traffic shaper computes $s(q, k + 1)$. The traffic shapers are presented in Section 2.3.2.

updateNextReceive(q,c, $r(q, k+1, c)$)) {
    Update the next expected receive time $q.r_{next}(c)$ with $r(q, k+1, c)$
    checkState()
}

updateNextSend(q,$s(q, k+1)$){
    Update the next expected send time $q.s_{next}$ with $s(q, k+1)$
    checkState()
}

checkState() {
    $t_{wakeup} = \min(\{t | t = q.s_{next} \ \ \forall q, c\} \bigcup \{t | t = q.r_{next}(c) \ \ \forall q\})$
    $t_{sleep} = t_{wakeup} - now$
    if ($t_{sleep} > t_{BE}$):
            sleep and set time to wake up at ($t_{sleep} - t_{OFF \rightarrow ON}$)
}

Figure 2.1: Safe Sleep Algorithm

Algorithm. SS works as follows. A node checks its state after it sends or receives a data report. Let t_{wakeup} be the minimum of the expected reception and send times of all queries. If t_{wakeup} is larger than the current time, then the node remains free until t_{wakeup}. Otherwise the node is busy since data reports are to be received or sent. If a node is free, SS may turn off its radio. However, to avoid incurring any delay or energy penalties the costs associated with transitioning between power states must be considered. To characterize these costs we define the break-even time t_{BE} as the minimum time the node needs to remain free such that there is no delay or energy penalty in turning the radio off and back on [11]. When the radio's transition power is no higher than its active power, the break-even time is the time it takes to transition from the active state to the off state ($t_{ON \rightarrow OFF}$) and back ($t_{OFF \rightarrow ON}$). A method for computing the break-even time when the transition power is higher than the active time is given in [11]. SS ensures that no energy or delay penalties are incurred through two steps: First, SS puts the node to sleep only if the nodes is free and remains free for longer than the break-even time. Second, the node sleeps until $t_{sleep} - t_{OFF \rightarrow ON}$ such that there is enough time to wake up. We call this algorithm Safe Sleep because it guarantees that no energy or delay penalties are incurred by turning the node off.

So far we have considered the operation of the system after the query setup is performed. The root starts a query by flooding a query request to all sources. During the setup slot, all nodes keep their radio on even if SS does not expect any data reports to be sent or received. For the duration of the setup slot both setup requests and data reports may be transmitted. Outside the setup slot, only data reports may be transmitted. The size of the setup slot affects both the time it takes to setup a query and the energy consumption at a node.

We note that SS has two notable features. First, it can flexibly schedule sleep periods for multiple queries. It does not need to maintain a TDMA schedule. Second, the storage cost of each query is proportional to the degree of the node in the routing tree. This localized property allows SS to scale effectively in large networks.

Impact of incorrect predictions. When the prediction of the workload is perfect, i.e., the *expected* reception time $r(q, k, c)$ and the *actual* reception times coincide, SS achieves the maximum sleep time. However, as discussed earlier, it is difficult to predict the actual reception time in WSNs due to delay jitter.

The accuracy of the expected reception time greatly affects the ability of SS to conserve energy. When the actual reception time is earlier than the expected reception time, a data report cannot be transmitted successfully because the receiver may be asleep. To avoid transmission failures, the traffic shapers always set the expected reception time of a child's data report to be the same as the child's expected send time of the same data report. Even if a data report is ready before its expected send time, the sender buffers it until the expected send time when the receiver wakes up.

Inaccuracies in the expected reception time lead to shorter sleep intervals because SS keeps the radio turned on from the time the data report is expected until the data report arrives. This situation has two possible causes. First, a child may not be able to send its data report to the MAC layer at the expected send time because it has not received the data reports from its own children due to communication delays. Second, even after a child sends the data report to its MAC layer, the data report suffers from additional delays due to queuing, contention, and transmission at the MAC layer.

2.3.2 Traffic shapers

In this subsection, we first analyze the performance of SS without traffic shaping and then present two traffic shapers that improve the energy efficiency with minimal delay penalty.

No Traffic Shaping (NTS)

In the following we describe how SS can be used without traffic shaping. In this case, SS only takes advantage of the periodicity of the data reports from the leaves of the routing tree. Since each node performs aggregation, a non-leaf node usually waits to receive data reports from its children before it transmits the aggregated data report. In NTS, a node sends its aggregated data reports to its parent *immediately* after it has received and aggregated the data reports from its children.

As mentioned earlier, even though data reports are produced periodically on each leaf node, the time when a node actually receives a data report is unpredictable due to accumulated jitter in multi-hop wireless communication. NTS estimates the reception times of data reports as follows. The reception time of the k^{th} data report of query q must be no earlier than the beginning of a query period $\phi + k * P$, where ϕ and P are the start time and the period of the query, respectively. [3] Every node shares the same expected send and reception times of the k^{th} data report: $s(k) = r(k) = \phi + k * P$. For a single query, a node turns on its radio at $r(k)$ and keeps listening until it receives all data reports from its children, performs the aggregation, and relays the aggregated data report to its parent. After the node sends the data report, it can turn off its radio until $r(k+1)$ when a new data report is produced. By design, SS also handles concurrent queries.

The protocol that employs SS without traffic shaping is denoted by NTS-SS. An advantage of NTS-SS is that it does not introduce any delay penalties. However, NTS-SS wastes energy by pessimistically turning on all nodes when a data report is generated on the leaf nodes.

[3] Previously the notation $r(q, k, c)$ was used to refer to the reception time of the k^{th} data report of query q from child c. When no ambiguity exists we will drop the variables q and c for brevity.

Analysis. We analyze the performance of NTS-SS in terms of energy efficiency and query latency. For clarity we only consider the case when a single query is registered. To evaluate the energy efficiency of our protocols we quantify the time a node is active. A node may be active because it expects data reports from its children or because it is sending data reports. The active time for sending a data report is not directly affected by the traffic shaper, making this case uninteresting for power analysis. Thus, we focus on analyzing the time a node remains awake to receive data reports from its children T_{recv}.

The energy conserved by a node using NTS-SS varies as a function of its *rank* in the tree. We define the rank d of a node to be the maximum hop count to any of its descendants in the routing tree. By definition a leaf node has a rank of zero.

For a leaf node, T_{recv} is zero since it does not receive data reports. The time a node with rank $d > 0$ remains active is the sum of three components: the maximum time it takes for each child to receive the data report $T_{recv}(d-1)$; the maximum time it takes its children to compute their data reports by applying the aggregate function T_{comp}; and the maximum time it takes to receive all data reports from its children $T_{collect}$. Therefore, $T_{recv}(d) = T_{recv}(d-1) + T_{collect} + T_{comp}$. Let T_{agg} be the upper bound on the time it takes for a node to receive all the data reports from all its children and generate a data report: $T_{agg} = T_{collect} + T_{comp}$. Accordingly, $T_{recv}(d)$ is:

$$T_{recv}(d) = \begin{cases} 0, & \text{if } d = 0; \\ (d-1) * T_{agg} + T_{collect}, & \text{if } d \neq 0. \end{cases} \tag{2.1}$$

Note that large variations in the energy conserved at different nodes limits the lifetime of the network. The nodes close to the root that have higher ranks will run out of energy faster than the others. Therefore, this NTS-SS exhibits good energy efficiency only when the routing tree is small.

We observe that the energy efficiency of NTS-SS may be improved by increasing the control over when the data reports are generated. This improves the accuracy of expected reception times at intermediary nodes, thus reducing the idle listening time. To this end, we introduce two traffic shapers.

Static Traffic Shaper (STS)

STS enforces the periodicity of data reports by pacing their multi-hop transmission over a period equal to an assigned *deadline* D. In our implementation, we allocate the same amount of time l to each rank in the tree . We let the local deadline l be: $l = \frac{D}{M}$ where M is the maximum rank of the tree.

For each query, the expected reception time of a node of rank d is $r(k) = \phi + k * P + l * (d-1)$ and its expected send time is $s(k) = \phi + k * P + l * d$. If a data report is generated before its expected send time $s(k)$ it is buffered until that time. If the data report is late, then the node sends it immediately. This mechanism reduces the difference between the expected and actual send and reception times and hence improves the energy efficiency of STS.

Analysis. Let STS-SS denote the ESSAT protocol that integrates STS and SS. We consider the case when a single query runs in the system. The critical parameter that must be tuned in STS is the local deadline l. The choice of l represents a tradeoff between the energy efficiency and query latency. We now analyze the impact of l on STS's query latency and the energy efficiency. Query latency is the maximum time it takes to send a data report from any source to the root. For STS, the query latency is:

$$L_q = M * max(l, T_{agg}) \tag{2.2}$$

When $l < T_{agg}$ a node with rank d expects to receive the data reports from its children at $l * (d-1)$ and turns on its radio at that time. However, since $l < T_{agg}$ the children cannot send the data reports on time. The data reports reach a node with rank d at $T_{agg} * (d-1) - T_{collect}$. Thus, $T_{recv} = T_{agg} * (d-1) - T_{collect} - l * (d-1) = (T_{agg} - l) * (d-1) + T_{collect}$. When $l \geq T_{agg}$ the time the node remains awake to receive the data reports is $T_{collect}$ since the children are ready to transmit their data reports in time. Therefore,

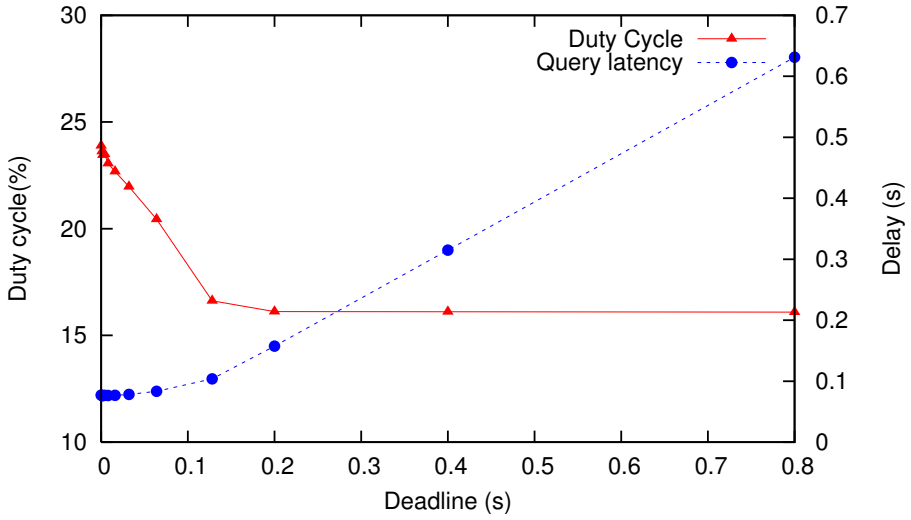


Figure 2.2: Impact of query deadline on duty cycle and query latency of STS-SS.

$$T_{recv}(l,d) = \begin{cases} 0, & \text{if } d = 0; \\ (T_{agg} - l) * (d-1) + T_{collect}, & \text{if } l \leq T_{agg} \wedge d \neq 0; \\ T_{collect}, & \text{if } l > T_{agg} \wedge d \neq 0. \end{cases} \tag{2.3}$$

Note that in the special case when $l = 0$, STS behaves like NTS.

Figure 2.2 shows the impact of the query deadline on the average duty cycle and query latency obtained in an *ns-2* simulation when three queries are running. The complete experimental setup is described in Section 2.4. The duty cycle of a node is defined as the percentage of time a node remains active during a query. A discontinuity point is observed in both average duty cycle and query latency at $D = 0.12s$ when the local deadline l approaches T_{agg}. When $D < 0.12s$, the query latency remains almost constant, while the average node duty cycle decreases monotonically as D increases. On the other hand, when $D > 0.12s$, the query latency increases proportionally with the deadline without reducing the average duty cycle. This result validates the above analysis.

STS-SS has maximum energy efficiency and good query latency when its local query deadline l approaches T_{agg}. However, due to the dynamic nature of WSNs it is difficult

to estimate T_{agg} accurately. Hence, it is difficult to tune the parameter l of STS. To address this limitation we develop the Dynamic Traffic Shaper.

Dynamic Traffic Shaper (DTS)

In contrast to STS which assigns fixed expected reception and send times, DTS dynamically adapts the expected send and reception times in response to variations in the multi-hop delays of received data reports.

Similarly to NTS, DTS initially sets the expected send and reception times to equal the start time of the query: $s(0) = r(0) = \phi$. Every time a node sends a new data report, DTS sets the expected send time of its next data report to $s(k) = s(k-1)+P$. Depending on whether or not the k^{th} data report is sent at the expected send time, DTS behaves differently. If a node receives the data reports from all of its children in time such that its k^{th} data report is ready before $s(k)$, it sends it at $s(k)$ and computes the next expected send time as $s(k+1) = s(k) + P$. After receiving the data report, the parent sets its next expected reception time to $r(k+1) = r(k) + P$. No explicit synchronization between a node and its parent is necessary in this case. However, when the k^{th} data report is ready at a time $t > s(k)$, DTS sends the data report immediately and sets the next expected send time $s(k+1) = t + P$. This case is called a *phase shift*. When a phase shift occurs, the node piggybacks $s(k+1)$ in the packet containing the data report. After receiving the data report, the parent sets its next expected reception time to $s(k+1)$.

We note that, the expected reception and send times are computed similarly to the Release Guard protocol[123] developed for multiprocessor real-time systems. However, Release Guard and DTS are designed for different purposes and systems. Release Guard is used to compute task release times to allow end-to-end schedulability analysis for real-time systems. In contrast, DTS is used as a traffic shaper for efficient sleep scheduling in WSNs. Unlike Release Guard that deals with chains of tasks, DTS handles data aggregation on multi-hop routing trees. Moreover, to improve energy efficiency, DTS re-synchronizes a child and its parent through an explicit packet exchange when data reports are dropped (discussed in Section 2.3.3) while Release

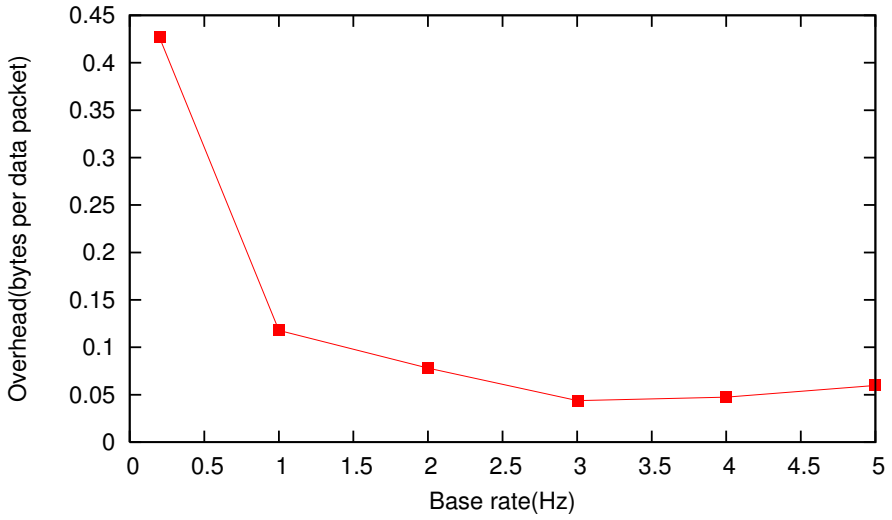


Figure 2.3: Overhead of DTS-SS in bytes per packet

Guard does not support this mechanism because it is not concerned with sleeping nodes.

Analysis. As described above, if a node does not receive a child's data report by the expected reception time, it wastes energy listening until the report arrives. To prevent future energy wastage, the node performs a phase shift to postpone its expected send time of the next data report from that child. Essentially, DTS *adapts* to the network workload by adjusting the expected send and reception times of data reports based on the longest multi-hop delay of received data reports. Once a node converges to T_{agg} no other phase shifts occur.

The adaptive feature of DTS is accomplished at the cost of additional synchronization overhead. However, since DTS advertises the expected send time of the next data report *only* when a phase shift occurs its communication overhead is small. To verify this conjecture, we run DTS in the presence of three queries with different rates under the experimental setup described in Section 2.4. On average the overhead due to piggybacked phase updates is less than *one bit per data report* for all tested query rates. (see Figure 2.3). The low overhead indicates that DTS is suitable for bandwidth-constrained WSNs.

Through protocol analysis we observed a tradeoff between energy conservation and query latency. Transmitting the data reports immediately, as in the case of NTS, does not incur any delay penalties, but this comes at the cost of poor energy efficiency. In contrast, STS and DTS conserve additional energy through traffic shaping at the cost of slightly increased delay. In STS, the parameter l controls the tradeoff between energy efficiency and query latency. Since tuning l is difficult, DTS has the practical advantage of being self-tuning.

2.3.3 Protocol Maintenance

A key feature of the ESSAT protocols is their robustness in face of network dynamics. We are interested in analyzing the behavior of the ESSAT protocols under two failure modes: when packets are transiently lost and when the topology changes due to persistent link and node failures.

Transient packet loss. In the case of transient packet loss, NTS-SS and STS-SS require no corrective action because the expected reception and send times are independent of the node's parent or child. In contrast, in DTS-SS, when a data report that contains a phase update is dropped the nodes become unsynchronized. When a packet loss is detected (e.g., based on the sequence numbers of received data reports), a node resynchronizes its schedule as follows. If the data report received after the transient packet drop(s) contains a phase update, this phase is used as the new phase for DTS-SS. Otherwise, the receiver requests a phase update from the sender. If the transmission of a data report is acknowledged, the receiver may piggyback the request for a phase update in the acknowledgement packet. Otherwise, a new packet is sent to request a phase update. The sender then piggybacks the expected send time in the next data report. Because a phase shift *only delays* the expected send time of future data reports, the receiver can tolerate the loss of multiple consecutive phase updates. However, this leads to transient energy waste because the node remains awake until the sleep schedules are resynchronized.

Topology changes. In the case of a persistent node or link failures the query service or routing protocol is responsible for reconfiguring the routing tree. When a node

fails both the parent and the children of the failed node need to recover from the failure.

A node discovers that it is the parent of a failed node if one of its children repeatedly fails to deliver its data report. In this case, all ESSAT protocols take two actions. First, the parent removes its dependency on the failed node, such that it no longer waits for data reports from a failed child. Second, the stale expected send and reception times of the failed node used by SS are removed.

A node discovers that it is the child of a failed node if it repeatedly fails to transmit its data report to its parent. The first step in the recovery process is for the query service or routing protocol to identify a new parent. The new parent adds a dependency on the node such that it generates aggregated data report after the child contributed its data report. The second step is to update (if necessary) the expected send and reception times of the node. NTS-SS does not require an update since all nodes share the expected send and reception times. In contrast, for STS-SS, $s(k)$ and $r(k)$ depend on the rank of the node. As such, when the parent is changed the rank of the node may also change. When the rank changes, the considered node and its descendants must recompute $s(k)$ and $r(k)$ according to their new rank in the tree. This may incur additional overhead. In the case of DTS-SS, when a node changes its parent, the expected send time and expected reception times are synchronized through one phase update when the node sends its first data report to the new parent. An advantage of DTS-SS is that it does not require any special mechanism for dealing with topology changes.

Selecting timeout values. In the case when packets are lost, either due to transient or permanent failures, a node may receive the data reports from a subset of its children. To avoid waiting indefinitely for the children's data reports, a parent times out and sends the aggregated data reports based on the ones it has received. For NTS-SS the timeout interval is set based on the node's rank: $t_{TO}(d) = (d+1) * \frac{D}{M}$. For STS-SS the timeout is set relative to the expected send time $s(k)+l-t_{TO}$, where l is the local deadline and t_{TO} is a constant. For DTS-SS, since the time it takes a node to collect data from its children usually depends on the one-hop delay, the timeout is set to $max_c(s(k,c)) + t_{TO}$, where t_{TO} is a tunable parameter. The above timeout

values are selected to balance the query latency and the effectiveness of in-network aggregation. A detailed discussion is omitted due to space limitations.

We observe that the traffic shaper affects the robustness of the protocols with respect to network dynamics. Since NTS-SS computes the expected reception and send times based on the properties of the query, (ϕ, P), which are independent of both the behavior of the neighbors and the tree topology, it is the most robust, i.e., it does not require any state update (except for the time out value discussed above) to deal with packet loss or topology changes. Since STS-SS computes the expected reception and send times based on a property of the tree topology, namely the rank of a node, it is not affected by transient loss of packets. However, in the case of a change in tree topology, reception and send times may need to be updated according to the new ranks. A benefit of DTS-SS is that it does not require any special mechanism for handling topology changes and requires only a new phase update to resynchronize the sleep schedules when data reports are transiently dropped.

2.4 Experiments

Through *ns-2* simulations we evaluate the ESSAT protocols along three dimensions: energy efficiency, query performance, and the impact of the radio's break-even-time on energy efficiency.

In our simulations 80 nodes are randomly distributed in an area of 500 x 500 m^2. The communication range is set to $125m$. IEEE 802.11b is used as the MAC protocol. The network bandwidth is 1Mbps. Each data report is encapsulated in a single packet of 52 bytes.

We simulate three types of queries with different rates. The ratio of the rates of the three query classes $Q_1 : Q_2 : Q_3$ is $6 : 3 : 2$. Q_1's rate is referred to as the *base rate*. When the base rate is varied, the rates of Q_2 and Q_3 also change proportionally. We vary the workload in two ways. First, there is a single query per class and the base rate is varied from $1Hz$ to $5Hz$. Second, the base rate is fixed at $0.2Hz$ while the number of queries per class is increased. Each query starts at a random time chosen between $0 - 10s$. All experiments last for $200s$.

The root of the routing tree is the node closest to the center of the area. The root initiates the construction of the routing tree by flooding a setup request. Each node may receive setup requests from multiple nodes and selects the node with the lowest level as its parent. The routing tree is setup before the start of the experiments and spans all nodes located within $300m$ from the root. Each node in the routing tree performs in-network aggregation. We assume that each aggregated data report fits in a single data packet. STS-SS's deadline is equal to its period.

For performance comparison we run several baselines: SYNC, SPAN and PSM. The SYNC protocol uses a fixed duty cycle, an approach adopted by synchronous wake up protocols [138]. All nodes share a synchronized periodic schedule. Each period includes fixed active and sleep windows. We configure SYNC to run at a duty cycle of 20% and a period of $0.2s$. We chose a duty cycle of 20% to approximate the duty cycles of the ESSAT protocols in the case of medium workload. The period is set to be 0.2 seconds to coincide with the highest data rate ($5Hz$) of our experiments. We also chose PSM with the extensions proposed in [20] because it adapts to observed traffic through traffic advertisements. The beacon period, the ATIM window, and Advertisement window are set to $0.2s$, $0.025s$ and $0.1s$ respectively. SPAN [20] is a power management protocol that uses a communication backbone. To reduce query latencies, the routing trees are modified such that all leaf nodes are sleeping nodes while non-leaf nodes are active nodes selected by SPAN. In the original SPAN protocol non-active nodes run PSM. In our experiments, the leaf nodes run NTS instead of PSM since it has better energy performance and lower query latency than PSM.

Unless mentioned otherwise, each data point is the average over five runs. The start time of each query and the node locations are varied in each run.

2.4.1 Energy Efficiency

We use average node duty cycle as a metric to evaluate the energy efficiency of the considered protocols. Figure 2.4 shows the impact of increasing the base rate of the queries on the average duty cycle of all nodes. For this graph, the 90% confidence intervals of all protocols are within $\pm 2.3\%$. SYNC is not shown in Figures 2.4 and 2.5 because every node is configured to a fixed duty cycle of 20% for all experimental

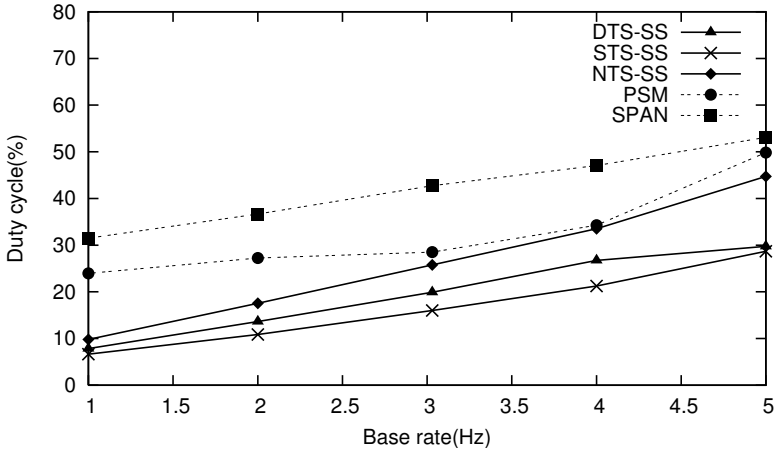


Figure 2.4: Average duty cycle for three queries classes when varying base rate.

settings. SPAN has the highest duty cycle in the network due to the high energy cost of maintaining a communication backbone. Since PSM does not maintain a communication backbone, it conserves more energy than SPAN. However, PSM transmits only overhead packets during the advertisement window incurring significant energy penalty. As a result, all ESSAT protocols have lower duty cycles than PSM. NTS-SS performs the worst among the ESSAT protocols. STS-SS has the best energy efficiency among the proposed protocols because its local deadline is longer than T_{agg} (see Equation 2.3). DTS-SS has a slightly higher duty cycle than STS-SS but remains consistently lower than NTS-SS. As the rate increases both STS-SS and DTS-SS increase their duty cycle to preserve the query performance (as shown in Section 2.4.2). These results are consistent with our analysis.

Figure 2.5 shows the average duty cycle when the base rate is $0.2Hz$ and the number of queries per class is increased. For this graph, the 90% confidence intervals of all protocols are within $\pm 1.2\%$. All ESSAT protocols again outperform the baselines. We note that DTS performs better in this experiment. Both Figures 2.4 and 2.5 show that DTS can effectively adapt to the workload without tuning.

As described in Section 2.3.1, a limitation of NTS-SS is that it consumes energy unevenly. To validate our analysis Figure 2.6 plots the average duty cycles of the nodes with the same rank. The plot shows the experimental results from a typical run when each class has one query and the base rate is $5Hz$. As the node rank increases, there is a linear increase in the duty cycle. This is consistent with our

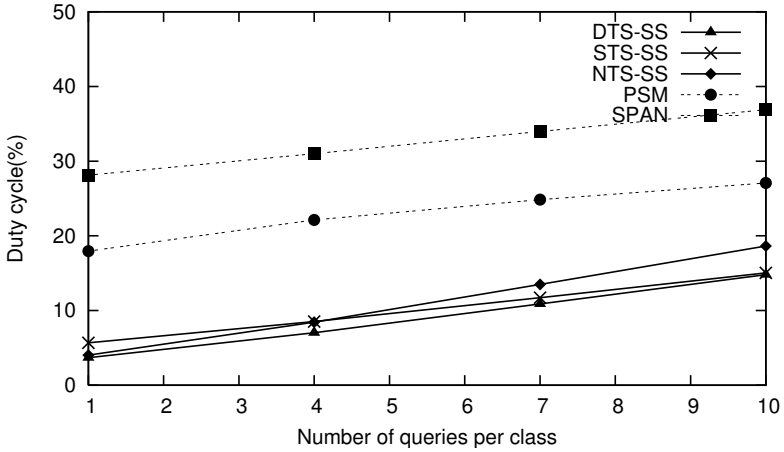


Figure 2.5: Average duty cycle for three query classes when varying number of queries per class.

analysis in Section 2.3.2. In contrast, the duty cycle of STS-SS and DTS-SS are independent of the rank of the node, making them more scalable. This result shows that STS-SS and DTS-SS distribute the energy consumption more evenly and can scale better to large routing trees than NTS-SS.

2.4.2 Query Performance

Figure 2.7 shows the average query latency as the base rate is increased. The 90% confidence intervals of ESSAT protocols and SPAN are within $\pm 0.16s$ while for SMAC and PSM the 90% confidence intervals are within $\pm 0.7s$. NTS-SS and SPAN have the lowest query latencies. Since NTS-SS propagates the data reports greedily and wakes nodes up in time, it does not introduce any delay penalties. SPAN achieves small query latency by maintaining an active communication backbone. However, as shown in Section 2.4.1, the query latency of SPAN comes at a high energy cost. All ESSAT protocols have significantly lower query latencies than SYNC and PSM. (Note the logarithmic scale of the figures). In SYNC and PSM the query latency is affected by the temporal relationship between the communication workload and their periodic sleep schedule. It is common for a data report to be buffered for considerable amount time resulting in higher query latencies. This problem is also reflected by the higher confidence of SMAC and PSM compared to that of the other considered protocols.

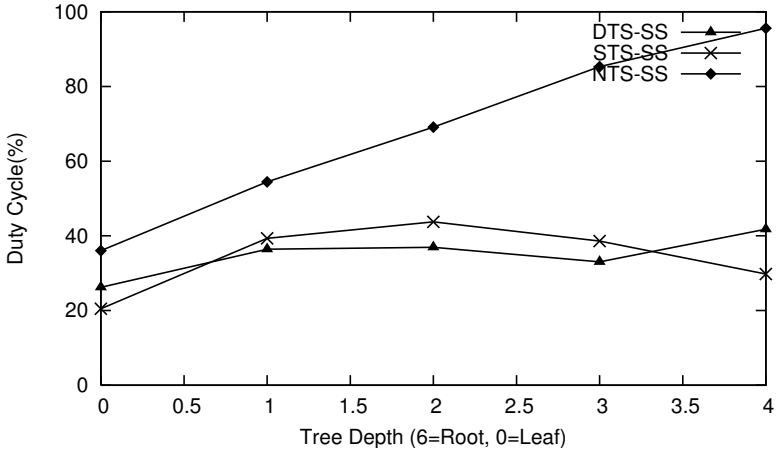


Figure 2.6: Distribution of duty cycles at different ranks.

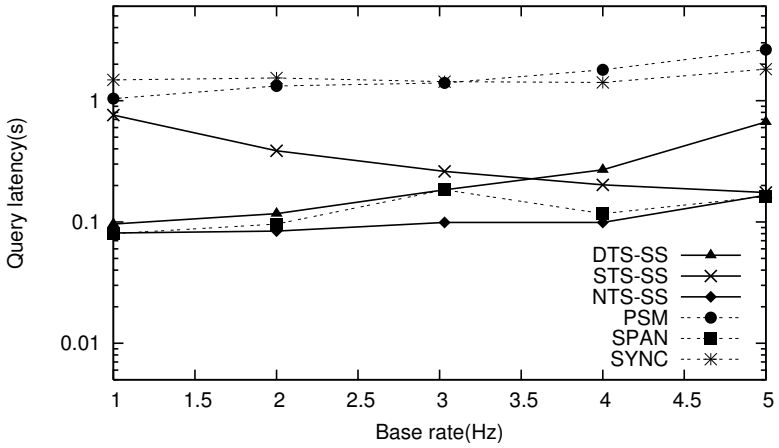


Figure 2.7: Query latency for three query classes when varying base rate.

As the local deadline of STS-SS is configured to equal the period of the query, its query latency decreases as the rate increases. In contrast, DTS-SS's query latency increases with rates. Despite this increase, the query latencies of DTS-SS are 36-98% lower than PSM and SYNC.

Figure 2.8 shows the query latency when the base rate is kept constant at $0.2Hz$ and the number of queries per class is increased. The confidence intervals are similar to the previously discussed experiment. In contrast with the previous setup, the latency of the STS-SS is constant since the rate does not change. DTS-SS exhibits lower query latency than STS-SS.

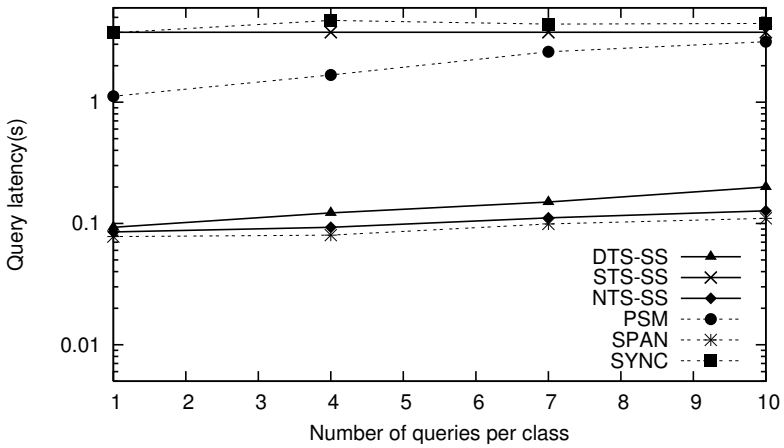


Figure 2.8: Query latency for three query classes when varying the number of queries per class.

2.4.3 Impact of Break-Even-Time of Radio.

As discussed in Section 2.3.1, fine grained power management needs to consider the costs of transitioning between power states. To quantify the impact of the break-even-time T_{BE} on energy efficiency of ESSAT protocols, Figure 2.9 plots the histogram of the sleep intervals when $T_{BE} = 0$. In light of Figure 2.9 we make two observations. First, based on the distribution, it is clear that the workload witnessed by nodes is aperiodic. Second, the impact of break-even-times must be taken into account. Otherwise a node suffers high penalties in query latency. For example, the percent of sleep intervals shorter than a break-even-time of $2.5ms$ (typical wake up delay for MICA2's radio and WLAN) for NTS-SS, STS-SS and DTS-SS are 0.40%, 0.85%, and 6.33% respectively. This implies, that for DTS-SS, 6.33% of the times the node is turned off additional delay penalties would have occurred if the break-even-times had not been taken into consideration. A key advantage of SS is that it avoids such delay penalties as discussed in 2.3.1. Since DTS-SS is the most sensitive to break-even-times we plot duty cycles of DTS-SS when different T_{BE} values are used as parameters for SS. We carefully chose the values of T_{BE}. The 2.5ms and 10ms are the average and worst case break-even-times reported for MICA2's radio [101]. The T_{BE} of 40ms is reported in ZebraNet [85]. For T_{BE} values smaller than 10ms which are common to MICA2 motes, the duty cycle is increased by at most 10%. However,

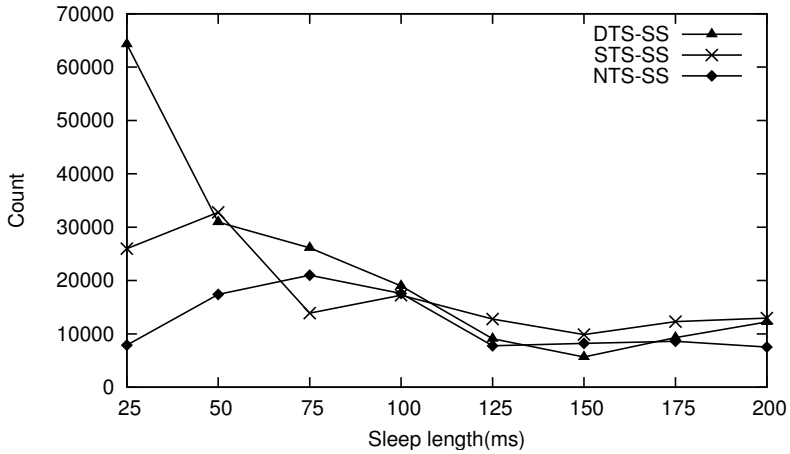


Figure 2.9: Histogram of sleep intervals. Each point in the graph represents the number of sleep intervals whose length falls in the range $[x-25, x]ms$.

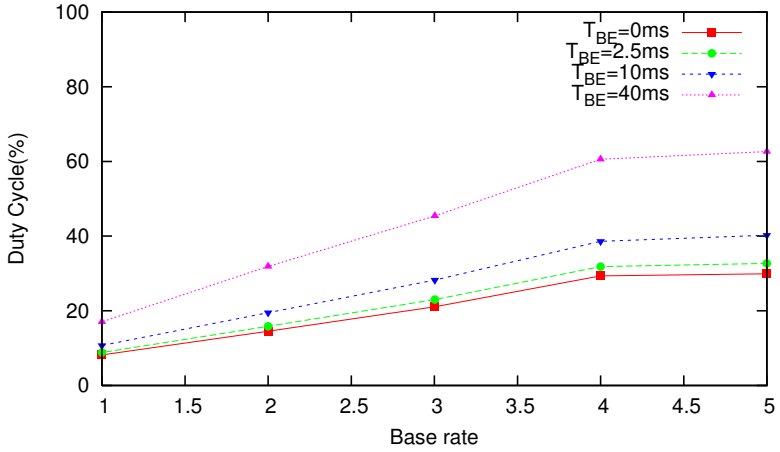


Figure 2.10: Impact of distribution of sleep interval on STS-SS. Three queries are run at a base rate of $5Hz$.

for $T_{BE} = 40ms$, an increase of as high as 30% is observed. This results confirm the importance of reducing the wake up time of radios in WSNs [101].

2.5 Summary

This chapter presents ESSAT, a novel power management scheme that aggressively exploits the timing semantics of WSN applications. An ESSAT protocol is comprised

of the Safe Sleep (SS) scheduler and traffic shaper (STS or DTS). A key feature of our ESSAT protocols is that they can conserve significant energy while introducing minimum delay penalties. For example, our simulations showed that DTS-SS achieved a average node duty cycles that are 38-87% lower than SPAN, and query latencies that are 36-98% lower than PSM and SYNC. Moreover, they can adapt to varying workload and network topologies at minimum overhead. As a result, ESSAT is especially suitable for WSNs that operate under both power and timing constraints in dynamic environments. In the future, we plan to integrate our protocols with existing query services on a testbed of MICA2 motes.

Chapter 3

Real-time Power-Aware Routing

Many wireless sensor network (WSN) applications require real-time communication. For example, a surveillance system needs to alert authorities of an intruder within a few seconds of detection [52]. Similarly, a fire-fighter may rely on timely temperature updates to remain aware of current fire conditions [87]. Supporting real-time communication in WSNs is very challenging. First, WSNs have lossy links that are greatly affected by environmental factors [140][18]. As a result, communication delays are highly unpredictable. Second, many WSN applications (e.g., border surveillance) must operate for months without wired power supplies. Therefore, WSNs must meet the delay requirements at minimum energy cost. Third, different packets may have different delay requirements. For instance, authorities need to be notified sooner about high-speed motor vehicles than slow-moving pedestrians. To support such applications, a real-time communication protocol must adapt its behavior based on packet deadlines. Finally, due to the resource constraints of WSN platforms, a WSN protocol should introduce minimal overhead in terms of communication and energy consumption and use only a fraction of the available memory for its state. To address these challenges, we propose the *Real-time Power-Aware Routing (RPAR)* protocol, which supports energy-efficient real-time communication in WSNs. RPAR achieves this by dynamically adapting transmission power and routing decisions based on packet deadlines. RPAR has several salient features. First, it improves the number of packets meeting their deadlines at low energy cost. Second, it has an efficient neighborhood manager that quickly discovers forwarding choices (pairs of a neighbor and a transmission power) that meet packet deadlines while introducing low communication and energy overhead. Moreover, RPAR addresses important practical issues in WSNs, including lossy links, scalability, and severe memory and bandwidth

constraints. In the rest of the chapter, we first analyze the impact of transmission power on communication delay via an empirical study (Section 3.1) and identify the design goals for real-time power-aware routing (Section 3.2). Next, we present the design of RPAR (Section 3.3). We evaluate the performance of RPAR through simulations based on a realistic radio model (Section 3.4). We conclude the chapter with discussions on open issues (Section 3.5) and related work (Section 3.6).

3.1 Impact of Transmission Power on Delay

RPAR is based on the hypothesis that there is an inherent tradeoff between transmission power and communication delay. In this section, we study the impact of transmission power on communication delay in WSNs. We first quantify their relationship through experiments on XSM2 motes. We then discuss the tradeoff between communication delay and network capacity.

3.1.1 Empirical Study on XSM2 Motes

To understand the impact of transmission power on end-to-end communication delay, we perform a set of experiments in an office environment using XSM2 motes. Each XSM2 mote is equipped with a Chipcon CC1000 radio. The bandwidth of the radio is 38.4 Kbps, but the effective bandwidth is lower due to packet loss. Five XSM2 motes are placed in a line. The first mote injects packets into the network at a rate of 4 packets per second. Each mote forwards a packet to its next neighbor. When a packet reaches the end of the line, the last mote changes the packet's direction and sends it back to the source. Each mote runs TinyOS with B-MAC [99] as the MAC protocol. We implemented the Automatic Repeat Request (ARQ) mechanism to improve reliability. Each packet is transmitted at most 5 times. The data and acknowledgement packets are transmitted at the same transmission power. The transmission power is varied from -18 dbm to 2 dbm in increments of 1 dbm. The one-hop distance is varied from 5 feet to 40 feet, in increments of 5 feet. 100 packets are sent at each power level.

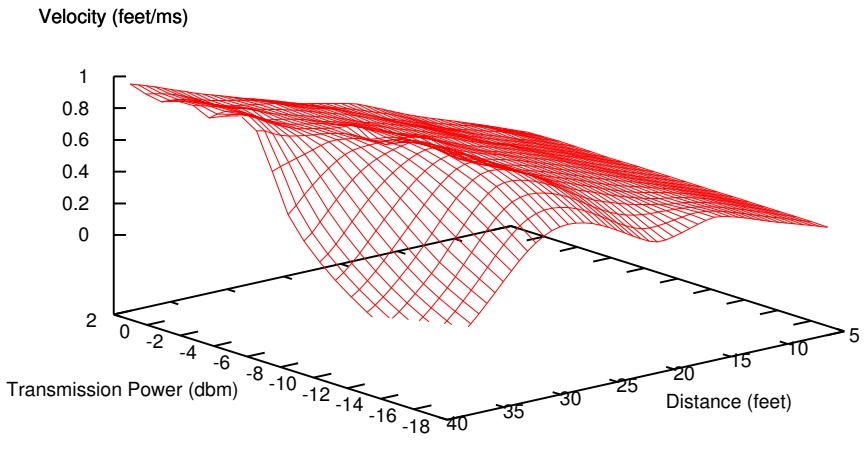


Figure 3.1: Impact of transmission power and one-hop distance on delivery velocity.

To evaluate the impact of transmission power on end-to-end delay, we measure the *delivery velocity* of each packet. The delivery velocity is defined as the distance a packet travels divided by its end-to-end delay. As shown in Figure 3.1, transmission power has a significant impact on delivery velocity. For example, when the one-hop distance is 20 feet, increasing the transmission power results in more than a two-fold improvement in delivery velocity, from 0.25 feet/ms at -18 dbm to 0.54 feet/ms at 2 dbm. This is because increasing transmission power effectively improves link quality [147] and, therefore, reduces the number of transmissions needed to deliver a packet. This shows that under light workloads, poor link quality is the root cause of long delays. At each power level, the delivery velocity increases as the one-hop distance increases within a range but drops sharply when the one-hop distance exceeds the range due to degrading link quality. The initial improvement in the delivery velocity is due to the packet traveling longer distances at each hop. The drop-off range of delivery velocity corresponds to the boundary of the gray area in packet reception ratio reported in [140][130]. A higher transmission power results in a longer drop-off range, e.g., the neighbor located at 40 feet is not in the communication range when the transmission power is -18 dbm but it has good link quality and high delivery velocity at 2 dbm. Therefore, distant nodes with poor quality links at low power are transformed into reliable communication neighbors when the transmission power is increased, achieving high delivery velocities.

Our experiments demonstrate that transmission power control may be an effective mechanism for controlling communication delays under light workloads. In the following subsection, we discuss the tradeoff between communication delays and network capacity under heavier workloads.

3.1.2 Tradeoff between Delay and Capacity

Increasing transmission power has the side effect of reducing the maximum achievable throughput in a WSN due to increased channel contention and interference [49]. Our focus is on real-time applications in which meeting the deadlines of critical data is more important than the total throughput. For example, in a surveillance application, timely delivery of the location of an intruder is more important to the user than delivering a large amount of non-critical data.

It is also important to note that the reduced capacity is a problem only when the workload approaches the network capacity. Recent advances in real-time capacity theory show that the performance degradation may be avoided as long as the amount of high-priority data transmitted in the network is small enough not to trigger capacity bottlenecks. In [3], the authors derive a lower bound on the maximum amount of real-time traffic that may be transmitted without triggering capacity bottlenecks. This bound may be used to perform off-line analysis of capacity requirements or, on-line, for admission control or congestion control. We discuss how to integrate RPAR with such techniques in Section 3.5.1.

RPAR achieves the desired tradeoff among communication delay, energy consumption, and network capacity by adapting the transmission power based on required communication delays. When deadlines are tight, RPAR trades capacity and energy for shorter communication delay by increasing the transmission power. Conversely, when the deadlines are loose, RPAR lowers the transmission power to increase throughput and reduce energy consumption. This adaptive approach is a key feature of RPAR.

3.2 Problem Formulation

Due to the unreliable and dynamic nature of WSNs, it is unrealistic to provide hard delay guarantees. RPAR assumes that each packet is assigned a soft deadline by the application, which specifies the desired bound on the end-to-end delay of a packet. The primary goal of RPAR is to increase the number of packets that meet their deadlines while minimizing the energy consumed for transmitting packets under their deadline constraints. RPAR focuses on minimizing the energy consumed in packet transmissions. In addition, RPAR is designed based on the following principles:

- WSN applications have varied communication requirements resulting in workloads with diverse deadlines. A real-time power-aware routing protocol should dynamically adapt its transmission power and routing decisions based on workload and packet deadlines.
- The design of RPAR should account for the realistic characteristics of WSNs including loss links [140] and extreme resource constraints in terms of memory, bandwidth and energy.
- RPAR should be localized protocol that makes decisions based solely on one-hop neighborhood information. This property enables RPAR to scale effectively to large WSNs.

In this chapter, we assume that each node is stationary and knows its location via GPS or other localization services [53]. Localization is a basic service essential to many applications that need to know the physical location of sensor readings. We also assume that radios can adjust their transmission power. For example, the Chipcon CC1000 radio can vary its transmission power between -20 dbm and 10 dbm. RPAR is designed to work with existing simple CSMA/CA protocols such as the B-MAC protocol [99] in TinyOS. To be consistent with the default configuration of B-MAC, RPAR assumes that the MAC protocol does not use RTS/CTS. RPAR may be easily extended to work with MACs that use RTS/CTS.

3.3 Design of RPAR

RPAR has four components: a dynamic velocity assignment policy, a delay estimator, a forwarding policy, and a neighborhood manager. RPAR uses the velocity assignment policy to map a packet's deadline to a required velocity. The delay estimator evaluates the one-hop delay of each *forwarding choice* (N, p) in the neighbor table, i.e. the time it takes a node to deliver a packet to neighbor N at power level p. Based on the velocity requirement and the information provided by the delay estimator, RPAR forwards the packet using the most energy-efficient forwarding choice in its neighborhood table that meets the required velocity. When the forwarding policy cannot find a forwarding choice that meets the required velocity in the neighbor table, the neighborhood manager attempts to find a new forwarding choice that meets the required velocity through power adaptation and neighbor discovery.

3.3.1 Dynamic Velocity Assignment Policy

Before a node S forwards a packet, it uses the velocity assignment policy to compute the *required velocity* based on the progress made toward the destination and the packet's *slack*. The slack is the time remaining until the packet deadline expires and is part of the packet header. The application running on the source node initializes the slack with the (relative) deadline. The slack is updated at each hop to account for the queueing, contention, and transmission delays. To measure the queuing delay node S time-stamps the packet when it is received ($t_{rec}(S)$) and when it becomes the head of the transmission queue ($t_{head}(S)$). Let $slack_{rec}(S)$ be the slack of the packet when S receives it. We account for the queueing delay by subtracting it from the slack, i.e., $slack(S) = slack_{rec}(S) - (t_{head}(S) - t_{rec}(S))$. The required velocity of a packet when it becomes the head of the transmission queue is:

$$v_{req}(S, D) = \frac{d(S, D)}{slack_{rec}(S) - (t_{head}(S) - t_{rec}(S))} \tag{3.1}$$

where, D is the destination of the packet and $d(S, D)$ is the Euclidean distance between S and D. It is important to note that the deadline is met if the required

velocity is met at each hop. Hence, RPAR maps the problem of meeting end-to-end deadlines to the local problem of meeting the required velocity at each hop.

When a node acquires the channel and is about to transmit a packet, it updates the slack in the packet header to account for the contention and transmission delays before transmitting it. The slack are also updated to account for the additional delays before each retransmission caused by packet loss. Note that updating the slack does not require clock synchronization.

This dynamic velocity assignment policy adapts the packet's required velocity based on current network conditions. If a packet is late, then its required velocity is increased so that it may catch up. Conversely, if the packet is early, its required velocity is decreased. In addition, the velocity assignment policy is used to prioritize the packets in the transmission queue according to their required velocity, with packets having a higher required velocity being transmitted first.

3.3.2 Forwarding Policy

RPAR makes forwarding decisions on packet-by-packet basis. In the following discussion we assume that RPAR forwards the packet that is the head of the transmission queue on node S and is destined for node D. RPAR forwards the packet to the most energy-efficient forwarding choice that meets the packet's required velocity. The velocity provided by (N, p) is:

$$v_{prov}(S, D, (N, p)) = \frac{d(S, D) - d(N, D)}{delay(S, (N, p))} \tag{3.2}$$

The progress made toward the destination by forwarding the packet to N is $d(S, D) - d(N, D)$. The estimated delay of forwarding choice (N, p), $delay(S, (N, p))$, approximates the time interval from when a packet becomes the head of the transmission queue until it is received at the next hop. The estimate is computed using the delay estimator (described in Section 3.3.3). Note that the delay estimator does not consider the queuing delay as it is already accounted for in the dynamic velocity assignment policy (as discussed in Section 3.3.1). The estimated one-hop delay is

used to determine if a forwarding choice is eligible. A forwarding choice (N, p) is an *eligible forwarding choice* if the velocity it provides $v_{prov}(S, D, (N, p))$ is higher than the packet's required velocity $v_{req}(S, D)$.

RPAR then estimates the energy cost of all eligible forwarding choices. It uses the following formula to approximate the energy consumption of routing a packet from the current node S to its destination D through forwarding choice (N, p):[4]

$$E(S, D, (N, p)) = E(p) \cdot R(S, (N, p)) \cdot \frac{d(S, D)}{d(S, D) - d(N, D)} \tag{3.3}$$

where $E(p)$ is the energy consumed for transmitting a packet at power level p. $R(S, (N, p))$ is the expected number of transmissions before S successfully delivers a packet to N when transmitting at power level p. R is computed by the delay estimator. $d(S, D) - d(N, D)$ represents the progress towards D when N is selected as next-hop.

3.3.3 Delay Estimator

The delay estimator is responsible for estimating the delay of different forwarding choices. The delay of a packet sent by S to neighbor N using transmission power p depends on the contention delay $delay_{cont}(S)$ (i.e., the time to acquire the channel), the total transmission time of the packet and its acknowledgement $delay_{tran}$, and the transmission count $R(S, (N, p))$[5]:

$$delay(S, (N, p)) = (delay_{cont}(S) + delay_{tran}) \cdot R(S, (N, p)) \tag{3.4}$$

Since the total transmission time of a packet and its acknowledgement is a constant determined by packet size and network bandwidth, the main function of the delay

[4]Equation (3.3) resembles the routing metric proposed in [111], which outperformed greedy geographic routing when a *fixed* transmission power is used. Our forwarding policy extends this metric to estimate the energy cost of forwarding choices with *different* power levels.

[5]In case of a failed transmission, the sender waits for the transmission time of acknowledgement before retransmitting the packet. The data and acknowledgment packets are sent at the same power level. The propagation delay is ignored, as WSN usually use short-range radios.

estimator is to predict the contention delay and the number of transmissions required to successfully forward a packet to a neighbor. Since the contention delay is independent of the forwarding choice when RTS/CTS is not used, our delay estimator consists of a single contention estimator *per node* and a transmission count estimate *per forwarding choice*. This reduces the storage cost of the delay estimator.

Our delay estimator is designed to support real-time communication in dynamic environments. Existing link estimators are designed to estimate the *average* link quality [130][51]. These approaches are ill-suited for real-time communication since routing decisions based on average delays may result in a large number of deadline misses due to high variability in communication delays. In contrast, our delay estimator adapts Jacobson's algorithm [62] to calculate conservative estimations of contention delays and transmission counts. Jacobson's algorithm was originally used in TCP to compute a retransmission timeout based on round-trip times between the source and the destination of a TCP flow. The retransmission timeout is computed by adding to the average round trip time its variation multiplied by a scaling term. We adopt Jacobson's algorithm because it considers both the average and variation of the estimated variable and, as a result, provides a better estimate of its worst-case value. This enables us to reduce the number of deadlines misses. Similarly, we compute a conservative estimate of the transmission count for each forwarding choice by considering the average and variation in the observed transmission count. However, if a packet is dropped after exceeding the allowed number of transmissions (according to ARQ), the transmission count estimate is set to infinity. A conservative estimate of the contention delay is also computed based on the average and variation of the observed contention delays. Equation (3.4) is used to estimate the delay of a packet by combining the estimated transmission count and estimated contention delay.

3.3.4 Neighborhood Manager

A key challenge for RPAR is to quickly discover eligible forwarding choices that are energy-efficient. This is particularly challenging because a node needs to select among a large number of forwarding candidates of which only a few may meet the velocity requirements. In WSN, due to the probabilistic nature of wireless links [130], a node hears transmissions from a potentially large number of neighbors including

those that have poor link quality and long delays. In addition, wireless interfaces typically have a large number of power levels to support fine-grained power control. For example, XSM2 motes support 31 power levels. Unfortunately, typical WSN platforms have limited memory allowing us to store only a few forwarding choices. For example, MICA2 motes have only 4KBs RAM, of which a small fraction may be dedicated to neighborhood management. More importantly, empirical studies show that link quality is highly variable over time [140][18][130]. As a result, the estimated delay and energy consumption of the forwarding choices that are used infrequently become outdated and inaccurate. Using this information may result in increased energy consumption or, even worse, in deadline misses. Therefore information about such forwarding choices must be refreshed when needed. This makes efficient discovery of eligible forwarding choices an important problem even on platforms without severe memory constraints.

RPAR features a novel neighborhood manager that dynamically discovers eligible forwarding choices and manages the neighborhood table. The neighborhood manager is invoked whenever there are no eligible forwarding choices in the neighbor table for forwarding a packet. It supports two mechanisms for discovering new forwarding choices: adapting the transmission power to a neighbor already present in the neighbor table (*power adaptation*) or discovering new neighbors (*neighbor discovery*).

Power Adaptation

When the required velocity of a packet cannot be satisfied, the power adaptation scheme increases the transmission power to improve the velocity provided by neighbors already in the neighbor table. Conversely, when velocity requirements are satisfied by known forwarding choices, it attempts to improve energy efficiency by decreasing the transmission power. RPAR adjusts the transmission power to a neighbor using a multiplicative increase and linear decrease scheme as discussed below.

When the forwarding policy cannot find an eligible forwarding choice in the neighbor table, RPAR determines a neighbor whose power should be increased to achieve higher delivery velocity. A node is *eligible for power increase* if its transmission count may be reduced by increasing the transmission power. A node becomes ineligible for

power increase if (1) the maximum transmission power is reached or (2) the estimated transmission count is one (i.e., the link quality is perfect). RPAR chooses the neighbor with the maximum velocity among all neighbors eligible for power increase, and multiplies the transmission power to that neighbor by a factor α ($\alpha > 1$). If RPAR cannot find a neighbor eligible for power increase, it invokes neighbor discovery to find new neighbors (see Section 3.3.4).

The power adaptation scheme may also decrease the transmission power to improve energy efficiency and network capacity. When the neighbor table contains forwarding choices that meet the velocity requirement of incoming packets, RPAR decreases the power of the most energy-efficient forwarding choice by β ($\beta > 0$) until one of the following conditions is satisfied: (1) the minimum power has been reached, (2) the estimated transmission count exceeds the number of allowed retransmission, or (3) there are two consecutive power levels such that at the lower level the required velocity is not met but at the higher power level the required velocity is met.

A large α reduces the time it takes to reach sufficient power to find an eligible forwarding choice. However, it may waste energy when a lower transmission power may suffice for meeting the required velocity. A large β allows RPAR to quickly reduce the power but it may also result in deadline misses when the power is reduced too aggressively.

The power adaptation scheme provides a responsive mechanism for adapting to variations in link quality. A key benefit of this scheme is that it does not introduce any overhead packets.

Neighbor Discovery

When RPAR cannot find an eligible forwarding choice through power adaptation, it uses the neighbor discovery component to find new neighbors that can meet the required velocity.

In the following discussion, we assume that S routes a packet to D and no eligible forwarding choice that meets the required velocity v_{req} exists in S's neighbor table. S starts neighbor discovery by broadcasting a Request To Route (RTR) packet at some

power p. Some node N hears the RTR and replies. Upon receiving the reply, node S inserts in its neighbor table the new forwarding choice (N, p). We need to address three issues: (1) What is the transmission power p at which an RTR is transmitted? (2) How can we minimize the communication overhead for neighborhood discovery? (3) How can we reduce the time it takes to find a neighbor that meets the required velocity?

When neighbor discovery is triggered because there is no neighbor closer to the destination in the neighbor table, S broadcasts an RTR at the medium power level. This usually occurs when a node routes a packet to a new destination. We chose to transmit the RTR at the medium power level to reduce the impact of neighbor discovery on network capacity and energy consumption. In contrast, if a neighbor closer to the destination is in the neighbor table, the RTR is broadcast at the maximum power. This ensures that far away nodes that may provide high delivery velocities receive the RTR.

Since the RTR is broadcast, a large number of nodes may reply and cause severe network contention. This problem can be mitigated by requiring each node to wait for a random interval before it is allowed to reply. A node does not reply if it hears replies from other nodes. However, This simple scheme has a drawback: while a large time window reduces the chance of packet collisions, it prolongs the time needed to find an eligible neighbor. It is crucial to discover new forwarding choices quickly, since neighbor discovery time is part of the end-to-end delay and therefore may lead to deadline misses[6].

To find a new neighbor, our neighborhood manager restricts the set of replying nodes to those that can meet the required velocity. A node replies only if it satisfies the following conditions: (1) it makes progress toward the destination, (2) it is not already in S's neighbor table, and (3) the maximum velocity that may be achieved by selecting it as next hop is higher than the required velocity. To verify that a node makes progress to the destination, we include the sender and destination locations in the RTR. In addition, the RTR may piggyback a list of node IDs which are in the table

[6]RPAR accounts for the delay caused by power adaptation and neighbor discovery by subtracting it from the slack.

and, hence, should not reply (within the limit of packet size). A neighbor N replies if the following inequality is satisfied:

$$v_{req}(S,D) \leq v_{max}(S,D,N) = \frac{d(S,D) - d(N,D)}{delay_{cont}(S) + delay_{tran}} \tag{3.5}$$

where $v_{max}(S,D,N)$ is the maximum velocity that N can provide to a packet being routed from S to D. v_{max} is computed based on the minimum possible delay which occurs when the transmission count between S and N is one ($R(S,(N,p)) = 1$). From (3.5), the maximum distance between any eligible neighbor N and destination D can be derived as follows:

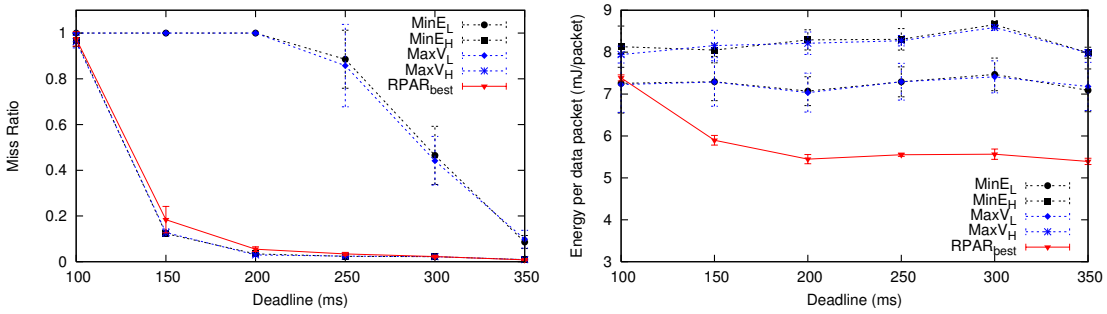
$$d_{max}(N,D) = d(S,D) - v_{req}(S,D) \cdot (delay_{cont}(S) + delay_{tran}) \tag{3.6}$$

S piggybacks $d_{max}(N,D)$ in the RTR, and a neighbor N that hears the RTR replies only if $d(N,D) \leq d_{max}(N,D)$.

Neighborhood Table Management

In contrast to earlier neighborhood management techniques that rely on periodic beacons [130], our power adaptation and neighbor discovery schemes are triggered *on-demand*, when no eligible forwarding choices exist in the neighbor table. The reactive approach enables our neighborhood manager to respond quickly to changes in the network conditions and packet deadlines while introducing low overhead when network and workload remain unchanged.

Similar to MT [130], we use the FREQUENCY algorithm [34] to manage the neighbor table. The FREQUENCY algorithm associates a frequency counter with each forwarding choice. When a forwarding choice is used for routing, its frequency counter is incremented while the frequency counters of all other forwarding choices are decremented. When the neighbor manager inserts a new forwarding choice and the table is full, the forwarding choice with the smallest frequency count is evicted. Since only the frequently used forwarding choices remain in the table, RPAR adapts the set of



(a) Miss ratio when the deadline is varied

(b) Energy per data packet when the deadline is varied

Figure 3.2: Performance of considered protocols when deadline is varied. The neighborhood table is prefilled.

known forwarding choices to the velocity requirements of the incoming packets. To avoid using stale data in larger neighbor tables, we add a timeout value to each forwarding choice which is reset upon using the forwarding choice. Forwarding choices that exceed the timeout value are considered stale and are evicted.

3.4 Experimental Evaluation

We implement RPAR in a Matlab-based network simulator called Prowler [115]. To create a realistic simulation environment, we configure Prowler based on the characteristics of MICA2 motes [1], which share the same hardware configurations as the XSM2 motes but with different packaging. A node can transmit packets at 31 power levels, ranging from -20 dBm to 10 dBm, with current consumption from 3.7 mA to 21.5 mA. The bandwidth is 40 Kbps. Prowler uses a log-normal shadowing path-loss propagation model at the physical layer. A collision occurs if a node receives two overlapping packets with signal strengths over the receiver's sensitivity. We implement the probabilistic link model from USC [144] in Prowler. Experimental data shows that the USC model produces lossy and asymmetric links similar to MICA2 motes [147]. The MAC protocol in Prowler employs a simple CSMA scheme similar to B-MAC, TinyOS's MAC protocol [99]. To improve the reliability, we use ARQ with a maximum number of five transmissions per packet. The size of the data and acknowledgment packets are 760 and 200 bits, respectively.

We evaluate RPAR's real-time performance and energy efficiency using the following performance metrics: *miss ratio*, defined as the fraction of packets that missed their deadlines, and the *energy per data packet*. The energy per packet is the total transmission energy consumed in a run divided by the number of data packets successfully delivered to their destinations. We compare RPAR with two protocols that consider real-time or energy-efficient communication. The first baseline protocol, MaxV, is inspired by SPEED [51], which supports soft real-time communication by enforcing a uniform delivery velocity across the network. However, to reduce the delay MaxV, always chooses the neighbor with the maximum velocity. The second baseline, MinE, is an energy-efficient geographic routing protocol that selects as next hop the most energy efficient forwarding choice according to Equation (3.3). This routing scheme significantly outperforms greedy geographic routing in terms of energy efficiency in lossy wireless networks [111]. Unlike RPAR, these baseline protocols operate at a fixed transmission power level. We use $protocol_L$ and $protocol_H$ to denote the protocol (MaxV or MinE) that operates at the default power (0 dBm) and the maximum power (10 dBm), respectively.

In simulations, we focus on the "many-to-one" traffic pattern that is common in WSN applications. In each simulation, 130 nodes are deployed in a 150 m $\times$ 150 m region divided into 11.5 m $\times$ 15 m grids. A node is randomly positioned in each grid. To increase the hop count between sources and the sink, we choose sources from the left-most grids of the topology. The sink is located in the middle of the right-most grids. A source sets the interval between sending two consecutive packets to be the sum of a constant (300ms) and a random value that follows an exponential distribution. We vary the mean of the exponential distribution to create different workloads. Each result is the average of five runs. The 90% confidence interval of each data point is also presented.

We start by evaluating the performance of the three forwarding policies when the neighborhood table of each node contains all forwarding choices. The link quality estimators are initialized according to the USC link model. This set of experiments is designed to quantify the best-case performance of the forwarding policies in the presence of perfect knowledge about the neighborhood and link qualities. Next, we consider the case when the neighborhood table has limited size and the link quality

of each forwarding choice is estimated on-line. Finally, we evaluate the impact of different workloads on RPAR.

3.4.1 Performance of Forwarding Policy

The first set of experiments uses a light workload generated by three sources. Each source sends on average a packet every 4 s. To evaluate RPAR's ability to adapt to different real-time delay requirements, we vary the packet deadline between 100 ms and 350 ms. Figure 3.2(a) shows the miss ratio as the deadline is varied. The forwarding policies that use the default transmission power, MinE_L and MaxV_L, start missing deadlines when the deadline is 350 ms. As the deadline decreases, they miss an increasing number of deadlines until 200 ms when none of the transmitted packets meet their deadline. In contrast, the baselines using the maximum transmission power, MinE_H and MaxV_H, have significantly lower miss ratios. This result confirms our observation (see Section 3.1) that using a high transmission power can effectively reduce communication delay.

However, Figure 3.2(b) shows that the baseline protocols using high transmission power consume significantly more energy per packet. In contrast, RPAR consistently achieves both desired real-time performance and energy efficiency under different deadlines. As shown in Figure 3.2(a), RPAR achieves miss ratios close to MinE_H and MaxV_H. At the same time, as shown in Figure 3.2(b), RPAR consumes less energy than MinE_L and MaxV_L for all deadlines except 100ms. This is because our forwarding policy selects more energy-efficient forwarding choices that still meet the delay requirements. Note the correlation between the energy consumption and the deadline: RPAR spends additional energy to meet tighter deadlines. This shows the desired trade-off between real-time performance and energy efficiency.

3.4.2 Performance with Neighborhood Management

This set of experiments is designed to evaluate the performance of the forwarding policies when using neighborhood management. In the following experiments, RPAR uses the neighbor discovery scheme described in Section 3.3.4. In our simulations, we

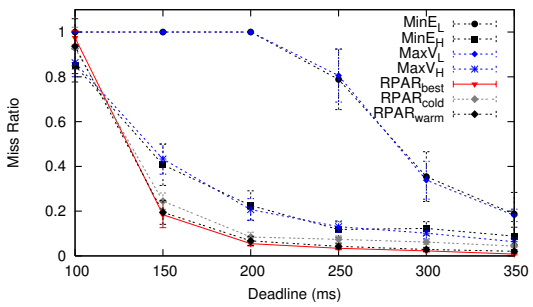
tune α so that it takes four iterations to increase the power from the default power level to the maximum power. We set $\beta = 1$. Similar to the MT protocol [130], the baselines use a neighborhood manager that uses beacons for neighborhood discovery and the FREQUENCY algorithm for table management. In all experiments, each node sends beacons with a period of 20 s using the same transmission power as the data packets. When the periodic beacon scheme is used, data packets start to be transmitted after 40 s to allow the link quality estimators in the neighborhood table to be initialized. The size of the neighbor table is set to 360 bytes for all protocols.

The performance of RPAR is affected by the quality of the forwarding choices found in the neighborhood table. As such, we consider three versions of RPAR. RPAR$_{best}$ quantifies the performance of the forwarding policy when the table is pre-filled, representing the best-case performance of RPAR. RPAR$_{cold}$ starts with an empty table and builds its neighborhood table according to the neighborhood management scheme described in Section 3.3.4. Since in practice the neighborhood table is usually not empty, RPAR$_{cold}$ represents the worst-case performance of RPAR. Therefore, we introduce RPAR$_{warm}$, which approximates the average-case performance when some forwarding choices are already in the table after routing the first 50 packets.

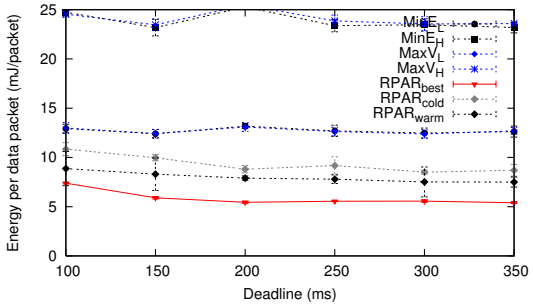
Figure 3.3 shows the performance of the forwarding policies used in combination with their respective neighborhood management policies. Figure 3.3(a) indicates that miss ratios of all protocols increased due to imperfect knowledge about forwarding choices. We observe a significant performance degradation (in terms miss ratio and energy consumption) when the beacon-based neighborhood manager is used with the baseline protocols whereas our on-demand neighborhood manager has a small impact on RPAR's performance. In contrast to the previous set of experiments where the baselines using the maximum transmission power had miss ratios comparable to those attained by RPAR, in these experiments RPAR clearly outperforms them. This shows that neighborhood management is an important issue in power-aware routing. The benefit of the new neighborhood manager is particularly evident for tight deadlines. At 150ms, the miss ratios of RPAR$_{cold}$ and RPAR$_{warm}$ are *only* 4.7% and 1.2% higher than RPAR$_{best}$, respectively. In contrast the miss ratio of MaxV$_H$ jumps up by 30.5%. Two factors contributed to the improved performance of RPAR's neighborhood discovery over the beacon based scheme. First, our neighborhood manager is deadline-aware in that it discovers and keeps forwarding choices that satisfy the

velocity requirement in the neighborhood table. Furthermore, our power adaptation and neighbor discovery schemes find good forwarding choices faster than the periodic beacons.

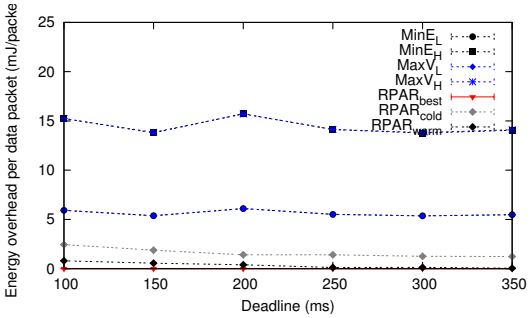
Figure 3.3(b) shows the energy consumed per data packet, including the energy spent for transmitting the overhead packets. Figure 3.3(c) shows the energy consumed for transmitting *only* the overhead packets. Figures 3.3(b) and 3.3(c) indicate that the energy consumed by the baseline protocols for neighborhood discovery accounts for a large part of the energy consumed per data packet. In contrast, RPAR which uses the on-demand neighborhood discovery scheme consumes significantly less energy. The reduction in energy consumption is attributed to both our forwarding policy (see Figure 3.2(b)) and our neighborhood manager which introduces significantly lower overhead. While the beacon period may be increased to lower the energy consumption, this would further degrade the real-time performance of the baselines.



(a) Miss ratio when deadline is varied



(b) Energy consumption per data packet when deadline is varied



(c) Overhead energy consumption per data packet when deadline is varied

Figure 3.3: Performance of considered protocols when deadline is varied (with neighborhood management).

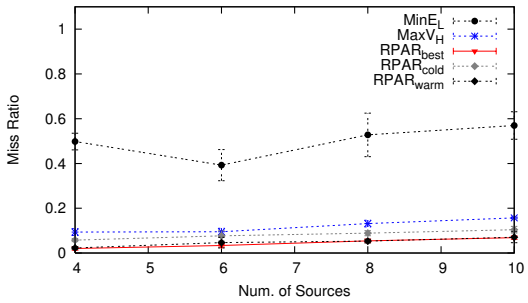
3.4.3 Impact of Workload

The final set of experiments evaluates the performance of RPAR under different workloads. Figures 3.4(a) and 3.4(b) show the experimental results for the case when the workload is varied by changing the number of sources from 4 to 10. Each source generates data with an inter-packet time of 6 s. The deadline is fixed at 300 ms. We observe that the number of deadline misses increases with the number of sources for all protocols except MinE$_L$. Because of the large confidence intervals of MinE$_L$, no clear correlation between its miss ratio and the increase in the number of sources may be established. The wide confidence intervals are the result of MinE$_L$ selecting unreliable links for transmission. The increase in the miss ratio with the number of source of the other protocols may be attributed to higher contention. The RPAR protocols have lower miss ratios and higher energy efficiency than the baselines.

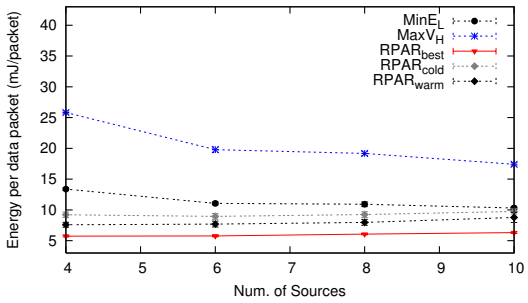
Figures 3.4(c) and 3.4(d) show the experimental results for the case when the average inter-packet time of each source is varied from 1s to 5s and the number of sources is fixed at 3. As the inter-arrival time is increased, the deadline miss ratios decrease for all protocols due to reduced network load. Similar to previous experiments, RPAR's miss ratio is lower than those of the baselines in all tested settings. Figure 3.4(d) indicates a increase in the total energy consumed by the baselines and a decrease in energy per data packet for RPAR. The increase in energy per data packet for the baselines is attributed to a lower number of dropped packets as the inter-packet packet time increases. RPAR incurs slight decrease in energy per data packet because it adaptively lowers transmission power under lower network loads. Overall RPAR consistently outperforms the baselines in term of energy efficiency when the inter-arrival time is 1 s.

3.5 Discussion

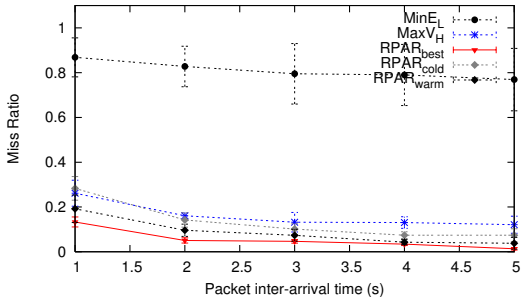
We now identify several open issues that have not been addressed in our current work and discuss how RPAR can be extended to address them.



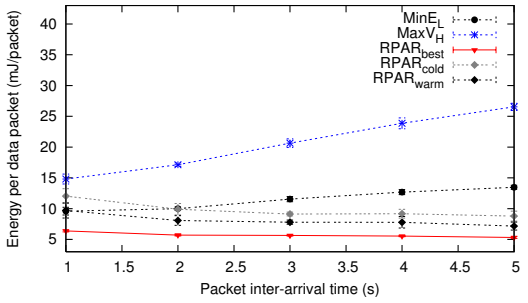
(a) Miss ratio when number of sources is varied



(b) Energy consumption per data packet when number of sources is varied



(c) Miss ratio when data rate is varied



(d) Energy consumption per data packet when data rate is varied

Figure 3.4: Performance of considered protocols when the workload is varied (with neighborhood management).

3.5.1 Handling Congestion

RPAR's power-adaptation policy exhibits a pathological behavior when a node is congested. Due to high contention, a node retransmits a packet several times before it is successfully received. Hence, RPAR increases the transmission power, which may further increase contention. RPAR can be integrated with existing protocols to deal with this problem. At the MAC layer, several methods have been proposed to differentiate between packets lost due to collisions or due to poor link quality [131]. Such feedback from the MAC layer would prevent RPAR from needlessly increasing the transmission power under congestion. Alternatively, RPAR may also work with existing congestion detection mechanisms for WSNs [57][126]. When a node detects congestion, RPAR should stop increasing the transmission power to avoid worsening the congestion and to allow the congestion control protocols [57][126][4] to alleviate it. Furthermore, RPAR may also be integrated with congestion and rate control techniques to keep the network below its real-time capacity bound [3].

3.5.2 Handling Holes

A known problem with greedy geographic forwarding is that it may fail to find a route in the presence of holes in the network topology. Such holes may appear due to voids in node deployment or node failures. RPAR partly mitigates this problem through power control: if the diameter of the hole is smaller than the transmission range at the maximum power, RPAR can transmit packets across the hole. For networks with large holes, RPAR needs to incorporate face routing mechanisms [66][13][76][70][39] to route packets around them.

When there are large holes, the Euclidean distance becomes a poor approximation of the actual path length. As a result, RPAR computes the expected number of hops incorrectly. We alleviate this problem through the dynamic velocity assignment policy, which recomputes the required velocity based on the progress toward the destination. The performance of RPAR may be further improved by considering the dilation of a path. This may be estimated by computing the boundary of a hole using a protocol such as BoundHole [39].

3.5.3 Integration with Power Management

RPAR aims to minimize the energy consumed for packet transmission. However, the cost of packet transmissions is only a part of the total energy consumption of a network. To further reduce total energy consumption, a WSN needs to integrate RPAR with a power-management protocol that reduces the energy wasted on idle listening. We consider two classes of power-management techniques and describe how RPAR may be integrated with them.

An effective power-management approach is to maintain a connected backbone composed of nodes that are always active, while the other nodes typically follow a periodic sleep schedule to save energy (e.g., [20][136]). The backbone is used for routing and buffering packets destined to sleeping nodes. The last-hop delay to a sleeping node is usually bounded by the period of its duty cycle and can be accounted for by adjusting the packet's velocity requirement.

Sleep scheduling algorithms alternate periods of sleep and activity. Of particular interest to real-time applications are sleep scheduling algorithms that adjust their periods of sleep and activity based on observed workload to minimize the impact of sleep schedules on message delay, such as T-MAC [125], 802.11 Power Saving Mode, ESSAT [27], on-demand power management [141], and the low-power listening scheme adopted by B-MAC [99]. As the packet is routed towards the destination, RPAR's dynamic deadline assignment policy can account for the additional delay introduced by sleep scheduling.

3.6 Related Work

RPAR is related to LAPC, SPEED, and MM-SPEED. LAPC [43] is a power-control protocol designed to reduce communication delays by adapting the transmission power to the workload. LAPC is not concerned with packet deadlines and only reduces communication delays in a best effort fashion. In contrast, SPEED, MM-SPEED, and RPAR are designed for real-time applications with explicit delay requirements.

SPEED [51] bounds the end-to-end communication delay by enforcing a uniform delivery velocity (called *speed* in [51] and [40]). MM-SPEED [40] extends SPEED to support different delivery velocities and levels of reliability. Both SPEED and MM-SPEED use fixed transmission power. In addition, RPAR differs from the above protocols in the following important respects. First, RPAR is the only protocol that integrates power control and real-time routing for supporting energy-efficient real-time communication. Furthermore, it allows the application to control the trade-off between energy consumption and communication delay by specifying packet deadlines. Second, unlike the other protocols, RPAR is designed to handle lossy links. This is an important feature since lossy links are common in WSNs and have a profound effect on communication delay [140][119]. Third, RPAR employs a novel neighborhood management mechanism that is more efficient than the periodic beacons scheme adopted by LAPC, SPEED and MM-SPEED. The simulations indicate that neighbor management has a significant impact on both real-time performance and energy efficiency.

There has been significant research on quality of service (QoS) support in wireless ad hoc networks. Several mechanisms to provide QoS support in 802.11 have been proposed. The most common approach is to provide service differentiation [46][2][64][60][95] by manipulating different MAC parameters. Overviews of these approaches are presented in [146] and [98]. Another approach for achieving QoS is to provide statistical guarantees on real-time traffic through online admission and rate control [4][10][137]. MAC layer prioritization, admission control and rate control may be used in conjunction with RPAR to further improve its real-time performance. Other authors propose routing protocols that provide QoS through path discovery and resource reservation [22][118] but none of the them use power control to achieve desired QoS.

Power-aware routing has been investigated in several previous works. For example, Singh et al. propose five power-based routing metrics that can be used to minimize power consumption or extend system lifetime [117]. Several power-aware protocols have been proposed to maximize network lifetime [82][19][109][19]. Power-aware routing has been implemented on real wireless network platforms [35]. Gomez and Campbell [48] provide theoretical analysis showing that allowing each node to dynamically adjusting its transmission power leads to improved capacity and energy-efficiency over

the case when all nodes use a common transmission power. Unlike RPAR, none of the above power-aware routing protocols is designed to support real-time communication.

3.7 Summary

We have developed RPAR, the first real-time power-aware routing protocol for WSNs. In contrast to existing protocols that treat real-time performance and energy efficiency in isolation, RPAR integrates novel real-time routing and dynamic power adaptation algorithms to achieve application-specified communication delays at low energy cost. Another distinguishing feature of RPAR is that it handles realistic properties of WSNs such as lossy links, limited memory, and bandwidth. Simulations based on a realistic radio model of MICA2 motes show that RPAR significantly reduces the deadline miss ratio and energy consumption compared with existing real-time and energy-efficient routing protocols.

Chapter 4

Dynamic Conflict-free Query Scheduling

Early research on wireless sensor networks (WSNs) has focused on low data rate applications such as habitat monitoring [93]. In contrast, recent years have seen the emergence of high data rate applications such as real-time structural health monitoring [24] and preventive equipment maintenance [75]. For instance, a structural health monitoring system may need to sample the acceleration of each sensor at rates as high as 500 Hz, resulting in high network load when a large number of sensors are deployed for fine-grained monitoring. Moreover, the system may have highly variable workload in response to environmental changes. For example, an earthquake may trigger a large number of new queries in order to assess any potential damage to the structure. Therefore, a key challenge is to provide a high-throughput query service that can collect data from large networks and adapt to workload changes.

To meet this challenge, we propose *Dynamic Conflict-free Query Scheduling* (DCQS), an integrated framework for transmission scheduling designed to meet the communication needs of high data rate applications. A data collection application may express its collection interests as queries over subsets of nodes which may involve data aggregation [89]. Instances of these queries are executed *periodically* to collect data at the base station. The use of routing trees in executing query instances introduces *precedence constraints* among packet transmissions. For example, when data aggregation is used, a node must wait for its children's data reports before computing an aggregated data report and relaying it to its parent. Intuitively, integrating application layer information (the periodicity of queries) and routing layer information (the

precedence constraints) into the transmission scheduling process may lead to significant performance improvements. By incorporating this cross-layer information into the scheduling process, DCQS provides not only better performance than traditional transmission scheduling techniques designed for general workloads and networks (see Section 4.4), but also has the following salient features:

- DCQS can dynamically adapt the transmission schedule in response to workload changes. As a result, queries may be added, removed, or their rates may be changed without having to recompute the transmission schedule.
- DCQS provides predictable performance in terms of maximum query throughput and power consumption. The predictability of DCQS enables it to effectively handle overload through simple rate control techniques and provide predictable network lifetime.
- DCQS has low run-time overhead and limited memory requirements making it suitable for resource constrained devices.

The remainder of the chapter is organized as follows. Section 4.1 describes the query and network models we adopt. Section 4.2 details the design and analysis of DCQS. Section 4.3 describes how DCQS handles dynamic networks and workloads. Section 4.4 provides simulation results using NS2. DCQS is compared with existing transmission scheduling approaches in Section 4.5. Section 4.6 concludes the chapter.

4.1 System Models

In the following we describe the query and networks models that DCQS builds upon.

4.1.1 Query Model

DCQS assumes a common query model in which source nodes produce data reports periodically. This model fits many applications that gather data from the environment

at user specified rates. Such applications generally rely on existing query services [90][134]. A query is characterized by the following parameters: a set of sources that respond to a query, the query period P_q, and the start time of the query ϕ_q, and an optional function for in-network aggregation [89]. *Query instances* are released periodically to gather data from the WSN. We use $I_{q,k}$ to denote the k^{th} instance of query q. The query instance $I_{q,k}$ is released at time $R_{q,k} = \phi_q + k \cdot P_q$ which we call the release time of $I_{q,k}$. For each query instance a node i needs $W_q[i]$ slots to transmit its (aggregated) data report to its parent. DCQS can support queries with in-network data aggregation, such as average and histogram [89], as well as more common forms of aggregation such as packet merging [100] and data compression [71], both of which can significantly reduce network load. While DCQS can optimize the performance of queries with aggregation, it can also support queries that do not perform aggregation.

A query service works as follows: a user issues a query to a sensor network through a base station, which disseminates the query description to all nodes. To facilitate data aggregation each non-leaf node waits to receive the data reports from its children, produces a new data report by aggregating its data with the children's data reports, and then sends it to its parent. We assume that there is a single routing tree that spans all nodes and it is used to execute all query instances. This assumption is consistent with the approach adopted by existing query services [89]. During the lifetime of the application the user may issue new queries, delete queries, or change the period of existing queries. DCQS is designed to support such workload dynamics efficiently.

4.1.2 Network Model

DCQS works by scheduling *conflict-free* transmissions in a time slot. To determine whether two transmissions are in conflict we introduce the Interference-Communication (IC) graph. The IC graph, IC(E,V), has all nodes as vertices and has two types of directed edges: *communication* and *interference* edges. A communication edge $\overrightarrow{ab}$ indicates that the packets transmitted by a may be received by b. A subset of the communication edges forms the routing tree that is used for data collection. An interference edge $\overrightarrow{ab}$ indicates that a's transmission interferes with any transmission

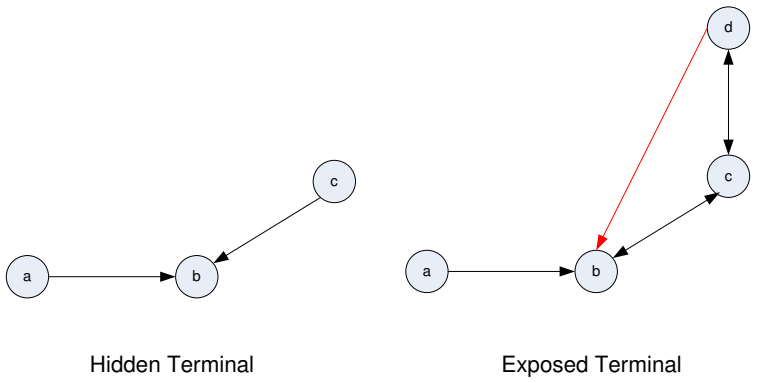


Figure 4.1: Conflicts in an IC graph

intended for b even though a's transmission may not be correctly received by b. An example of an IC graph is shown in Figure 4.3.

The IC graph is used to determine if two transmissions may be scheduled concurrently. We say that two transmissions, $\overrightarrow{ab}$ and $\overrightarrow{cd}$ are *conflict-free* ($\overrightarrow{ab} \parallel \overrightarrow{cd}$) and may be scheduled concurrently if (1) a, b, c, and d are distinct and (2) $\overrightarrow{ad}$ and $\overrightarrow{cb}$ are not edges in E. Referring to Figure 4.3, the transmissions $\overrightarrow{ea}$ and $\overrightarrow{fb}$ conflict due to the interference edge $\overrightarrow{eb}$. In contrast, transmissions $\overrightarrow{ne}$ and $\overrightarrow{po}$ are conflict-free, since edges $\overrightarrow{no}$ or $\overrightarrow{pe}$ are not part of the graph.

To illustrate the definition of conflict-free transmissions we consider the examples shown in Figure 4.1. In the first example, two transmissions $\overrightarrow{ab}$ and $\overrightarrow{cb}$ are scheduled concurrently. The two transmissions result in a collision at node b since condition (1) of the definition is violated. In the second example, $\overrightarrow{ab}$ and $\overrightarrow{cd}$ are scheduled concurrently. However, due to interference edge $\overrightarrow{cb}$, a collision occurs at b. This example is an instance of the hidden terminal problem and exemplifies condition (2) of the definition. The IC graph accounts for link asymmetry and for the irregular communication and interference ranges observed in WSN [143]. The IC graph may be stored in a distributed fashion: each node *only* needs to know its incoming/outgoing communication and interference edges.

RID, a realistic approach for constructing IC graphs based on Receive Signal Strength (RSS) measurements, is proposed in [143]. To gather RSS measurements, nodes

transmit sequences of two packets. The first packet is broadcast at maximum power and is used to identify the sender and prepare the potential interfering nodes to measure the RSS during the subsequent packet transmission. The second packet is transmitted at the default power level. Based on the collected RSS measurements, interference edges are added to the IC graph as follows. Consider a node p which receives packets from one of its children c. Node p knows c's RSS as well as the RSS of all other senders which may interfere with c's transmission. RID generates all sets of interferes $I(p)$ such that $|I(p)| \leq N_r$, where N_r is a bound on the number of senders that may be active in a time slot. Given the transmission $\overrightarrow{cp}$, RID computes SINR for each $I(p)$. If the SINR for the set of interferers $I(p)$ is below a threshold, then incoming edges from the node in $I(p)$ to p are added to the graph. The communication cost of RID is linear in the number of nodes.

The IC graph is based on the SNIR model. Empirical studies validating the accuracy of the SNIR model on 802.15.4 [92][113][143] and 802.11 [91] radios have already been performed. Moreover, MAC protocols which take advantage of the SNIR model have already been proposed and their performance validated empirically [113]. These previous studies on real hardware indicate that the IC graph is a realistic assumption. The IC graph was studied in [92], which presented a realistic approach for constructing IC graphs based on the SNIR model and RSS measurements. It should also be noted that the IC graph model adopted by our algorithm is significantly more realistic and general than those adopted by many earlier TDMA scheduling algorithms, such as circular models and those that ignore interference.

Interference is inherently probabilistic and changing over time. It is important to note that changing the SNIR threshold may control the temporal stability of the IC graph. A conservative SNIR threshold would lead to a more stable IC graph at the cost of reducing throughput. We recognize that even when using conservative SNIR threshold, packets may be still corrupted as a result of intermittent interference. As discussed in Section 4.3, we address these issues through packet retransmissions and multi-path routing.

While the IC graph is built conservatively to improve temporal stability, over time the interference relations may change significantly. We may detect changes in IC graph by monitoring the reliability of data collection over time. If the reliability falls below

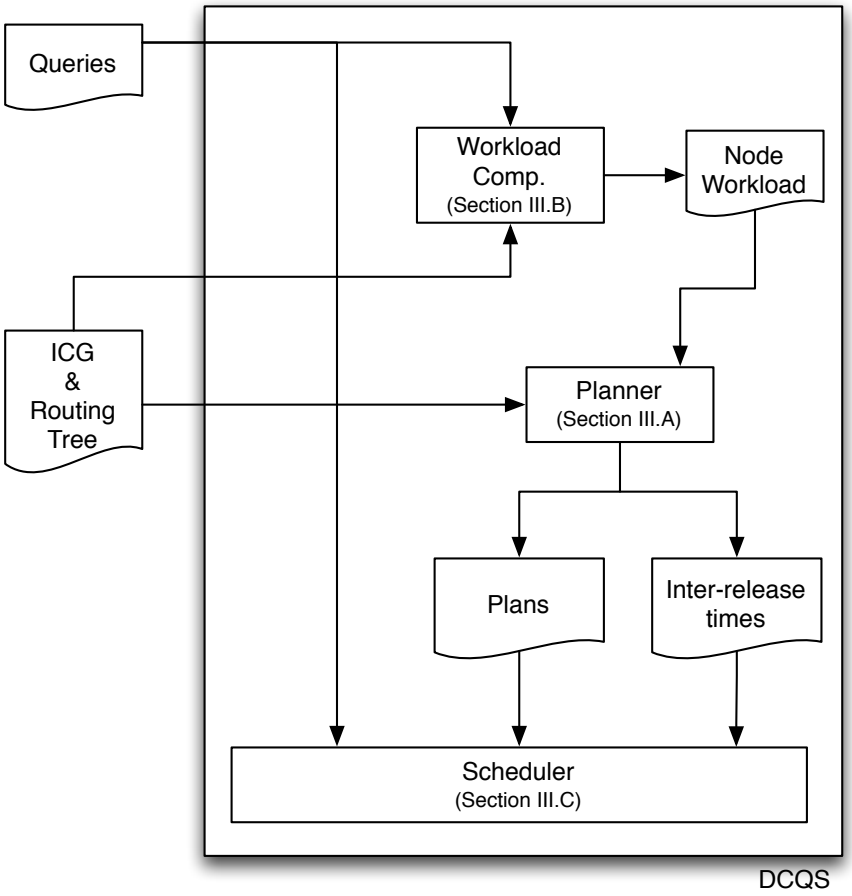


Figure 4.2: DCQS uses cross layer information in making scheduling decisions. It has two key components: a planner and a scheduler.

a user-set threshold then the IC graph is rebuilt. The IC graph may also need to be changed as nodes are added or removed on the graph. In the case of node additions or removal, the IC graph is rebuilt incrementally by running the RID protocols only on the newly added nodes. In the case of node removal, edges from the IC graph are removed locally.

4.2 Protocol Design

DCQS supports dynamic conflict-free query scheduling by separating the problem of transmissions scheduling into two parts. First, we consider the problem of scheduling

each query instance in isolation when all network resources are dedicated to its execution. To this end, DCQS constructs *plans* for executing each query instance. A plan is a sequence of *steps*, each comprised of a set of *conflict-free* packet transmissions which DCQS executes sequentially by performing the transmissions assigned to each step. Upon the completion of a plan execution, the data reports from all sources involved in the query would have been delivered to the base station.

Next, we consider the problem of executing a set of queries submitted by the user. DCQS could accomplish this by executing instances one at a time as they are released according to their previously constructed plans[7]. However, to improve throughput and reduce latency, DCQS dynamically determines which steps in the plans of the released query instances may be executed without conflict and executes them concurrently. Note that, unlike traditional TDMA protocols, DCQS does not maintain an explicit schedule but rather determines the schedule at run-time based on the temporal properties of queries and their plans. A significant contribution of this chapter is an efficient query scheduling algorithm designed for resource constraint devices.

This approach has several intrinsic advantages: (1) DCQS separates the costly process of constructing plans from the dynamic transmission scheduling performed in each slot. (2) To further reduce the overhead, DCQS reuses previously constructed plans for queries whenever possible. We will show that many queries may be executed according to the same plan. (3) The DCQS schedule executes query instances based on their temporal properties. Consequently, DCQS can handle changes in query rates and the addition/removal of queries efficiently. (4) Rate control may be performed at the base station to prevent overload.

To facilitate efficient query scheduling, DCQS shares information across the traditional protocol stack boundaries (see Figure 4.2). DCQS has two main components: a planner and a scheduler. The planner is responsible for constructing plans. The planner uses the IC graph and the following query information exposed by the application: the set of sources and the number of packets each node involved in a query has to transmit. The scheduler runs on every node and makes scheduling decisions at

[7]If this were the case, the network would execute a *single* instance at time even though some instances may be executed concurrently due to spatial reuse.

run-time based on the start time and period of queries as exposed by the application and the plans constructed by the planner.

DCQS works as follows: (1) When a new query is submitted, DCQS identifies a plan for its execution. As discussed in Section 4.2.1, it is often the case that many queries can be executed using the same plan. When no plan may be reused, the planner constructs a plan for executing the query. (2) Next, the base-station performs rate control to ensure that the total query rate remain within the maximum query rate under DCQS. If necessary, the rates of the queries are decreased proportionally to meet the maximum query rate (see Section 4.3.2). (3) The phase, period, and aggregation function of the query are disseminated to all nodes. (4) At run-time, the scheduler executes all query instances.

In the remainder of the section, we first present a centralized planner, which serves as a starting point for the design of the distributed algorithm. We then describe the local scheduler. The section concludes with the description of the distributed planner.

4.2.1 The Centralized Planner

In this section we present a centralized version of the planner.

Definitions. A *plan* is an ordered sequence of *steps* that executes a *query instance*.

A plan has the following properties: (1) In each step, conflict-free transmissions are assigned. (2) When the query involves aggregation, the plan must respect the precedence constraints introduced by aggregation: a node is assigned to transmit in a later step than any of its children. Note that DCQS does not impose any constraint on the order in which a node's children transmit. (3) Each node is assigned in sufficient steps to meet its workload demand. We use $T_q[s]$ to denote the set of transmissions assigned to step s in the plan of query q and L_q to denote the length of q's plan.

An example of a plan with seven steps is shown in Figure 4.4. In each step multiple conflict-free transmissions are assigned. For example, nodes n and p may transmit in step 2 since their transmissions do not conflict ($\overrightarrow{ne} \parallel \overrightarrow{po}$). The precedence constraints

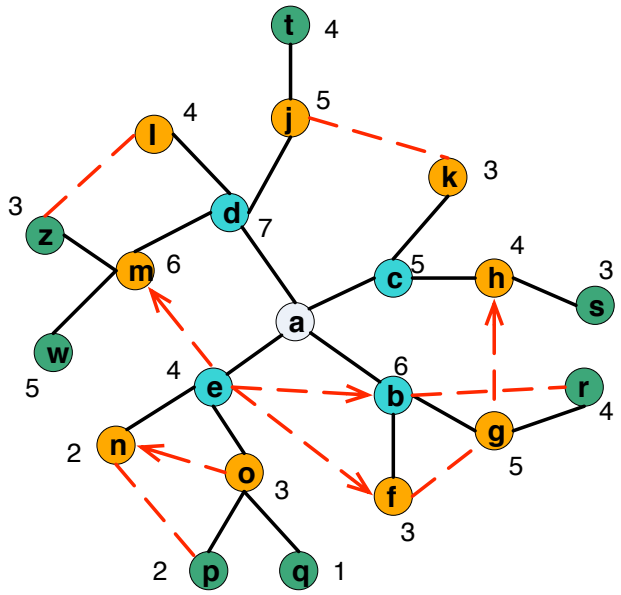


Figure 4.3: IC graph: The solid lines denote communication edges and form the routing tree. The dotted lines are interference edges. The edges without arrows are bi-directional. The shown numbers are the steps in which each node transmits under a plan for an instance with a workload demand of one slot per node.

	a	d	b	c	e	m	j	g	h	o	
1	d										7
2	b	m									6
3	c	j	g			w					5
4	e	l		h			t	r			4
5			f	k	o	z			s		3
6					n					p	2
7										q	1

Reversed Plan

Actual Plan

Figure 4.4: Constructed plan for IC graph in Figure 4.3 when each node has a workload demand of one slot. The first and last column are the step indices in the reverse and actual plans, respectively. The top row indicates the intended receivers. The entries in the other rows indicate the senders in each step of the plan.

introduced by aggregation are also respected: nodes p and q transmit in earlier steps than their parent o.

We opt for a node to wait for data from its children even for queries that do not involve aggregation because such an approach results in transmission schedules that have *long contiguous* periods of activity/inactivity: the node transitions from a sleep state to the active state just-in-time to receive the data from its children and transitions back to sleep after it completes collecting data from its children and relaying it to its parent. Such schedules are efficient because they reduce the wasted energy in transitions between sleep and active states.

Since a node waits to receive the data reports from its children (to support data aggregation and improve energy efficiency), the query latency may be reduced by assigning the transmissions of a node with a larger depth in the routing tree to an earlier step of the plan. This reduces query latency because it reduces the time a node waits to receive the data reports from all of its children.

The pseudo-code of the centralized planner is shown in Figure 4.5. The centralized planner works in two stages. In the first stage the planner constructs a *reversed* plan (V_q) in which a node's transmission is assigned to an *earlier* step than its children. In the second stage it constructs the actual plan (T_q) by reversing the order of the steps to enforce the precedence constraints. We will be using the notation $V_q[i]$ and $T_q[i]$ to refer to the set of transmissions assigned in ith step of the reverse and actual plans, respectively. The planner maintains two sets of nodes: *completed* and *eligible*. A node n is a member of the *completed* set if the planner has already assigned n to transmit in sufficient steps such that its workload demand is met. The *eligible* set contains nodes whose parents are in the *completed* set. Initially, the *completed* set contains the root of the routing tree and the *eligible* set contains its children. The planner considers the eligible nodes in order of their *priority* and assigns steps in which they transmit to their parents. The priority of a node depends on its depth, number of children, and ID. Nodes with smaller depth have a higher priority. Among the nodes with the same depth the ones with more children have higher priority. Node IDs are used to break ties. After the planner assigns steps for n to transmit to its parent, it moves n from the *eligible* set to the *completed* set, and adds n's children to the *eligible* set. The first stage is completed when the *completed* set contains all the nodes in

the network. In the second stage, the planner reverses the order of the steps in the reversed plan.

Let us consider how the scheduler assigns n's transmissions to its parent p in the reversed plan. The planner associates with each node two pieces of information: $n.minStep$ and $n.assignedSteps$. The value of $n.minStep$ is the step number in which the planner attempts to assign n's transmission to p, while the value of $n.assignedSteps$ is the number of steps in which n is assigned to transmit. Since nodes with smaller depth have a higher priority, p's transmissions to its parent has already been assigned to enough steps. Let s be the last step in the reversed plan V_q in which p transmits to its parent. In the reversed plan the earliest step in which n may transmit its own data report to p is $n.minStep = s + 1$. This means that, in the actual plan, p must transmit its data report to its parent at least one step before the parent transmits its data report such that the precedence constraints introduced by data aggregation are respected. To determine if the transmission $\overrightarrow{np}$ may be assigned to $V_q[n.minStep]$ without conflict, n must verify that all transmission pairs that involve $\overrightarrow{np}$ and any transmission already assigned to $V_q[n.minStep]$ are conflict free. The planner assigns node n to transmit in multiple steps until its workload demand $W_q[n]$ is met.

Figure 4.3 shows an example topology and the plan constructed by the centralized planner. All nodes have a workload demand of one slot. The constructed plan is shown in Figure 4.4. Initially, the children of the root a are eligible. The planner starts by scheduling d since it has the highest-priority among the eligible nodes. The planner assigns $\overrightarrow{da}$ to step 1 since $V_q[1] = \emptyset$. Next, b becomes the highest-priority eligible node. The first step in which $\overrightarrow{ba}$ may be assigned is step 1. However, since $\overrightarrow{ba} \nparallel \overrightarrow{da}$, $\overrightarrow{ba}$ cannot be assigned to that step. We assign $\overrightarrow{ba}$ to step 2 since $V_q[2] = \emptyset$. Similarly, $\overrightarrow{ca}$ and $\overrightarrow{ea}$ are assigned to steps 3 and 4, respectively. When the planner completes assigning e's transmission to its parent ($\overrightarrow{ea}$), m becomes the highest-priority eligible node. Since $\overrightarrow{da}$ is assigned to step 1, the first step to which $\overrightarrow{md}$ may be assigned is 2. Since in $V_q[2]$ only $\overrightarrow{ba}$ is assigned and $\overrightarrow{md} \parallel \overrightarrow{ba}$, we assign $\overrightarrow{md}$ to step 2. A more interesting case occurs when f becomes the highest-priority eligible node. The earliest step to which $\overrightarrow{fb}$ may be assigned is 3, since the transmission of its parent's transmission $\overrightarrow{ba}$ is assigned to step 2. The planner first attempts to assign $\overrightarrow{fb}$ to steps 3 and 4, but fails. $\overrightarrow{fb}$ cannot be assigned to step 3 due to $\overrightarrow{gb}$. $\overrightarrow{fb}$ cannot be assigned to step 4 because $\overrightarrow{ea} \nparallel \overrightarrow{fb}$ due to the interference edge $\overrightarrow{eb}$. Since no transmission

centralized-planner:
1: $completed = \{root\}$; $eligible = children(root)$;
2: while $(completed \neq V_q)$
3: Let n be the highest-priority node in $eligible$
4: invoke **assign-steps**(n)
5: $completed = completed \cup \{n\}$
6: $eligible = eligible \cup \text{children}(n)$
7: reverse plan: $T_q[s] = V_q[L_q - s]$

assign-steps(n):
9: Let p be n's parent and assigned.
10: Let $V_q[s]$ be the last step in which a transmission $\overrightarrow{np}$ is assigned
11: $n.minStep = s + 1; n.assignedSteps = 0$
12: while $(n.assignedSteps < W_q[n])$
13: if $\overrightarrow{np}$ does not conflict with any transmission $\overrightarrow{ab} \in V_q[n.minStep]$
14: $V_q[n.minStep] = V_q[n.minStep] \cup \{\overrightarrow{np}\}$
15: $n.assignedSteps = n.assignedSteps + 1$
16: else$n.minStep = n.minStep + 1$

Figure 4.5: The centralized planner.

is currently assigned to $V_q[5]$, $\overrightarrow{fb}$ is assigned to it. The first stage of the planner continues to produce the plan shown in the table. In the second stage, the planner reverses the order in which the steps are executed. Accordingly, the last step in the reversed plan ($V_q[7]$) is the first step in the plan ($T_q[1]$), the second to last step in the reversed plan ($V_q[6]$) is the second step in the plan ($T_q[2]$), and so on.

4.2.2 Plan Sharing

The plan of a query q depends on the IC graph, the set of source nodes, and the aggregation function. Queries instances executed at different times may need different plans if the IC graph changes. However, to handle dynamics in channel conditions, DCQS can construct plans that are robust against certain variations in the IC graph (as discussed in Section 4.3.3). This allows instances executed at different times to be executed according to the same plan. More importantly, note that queries with the

same aggregation function and sources but with different temporal properties (i.e., period, start time) can be executed according to the same plan.

Even queries with different aggregation functions may be executed according to the same plan. Let $W_q[i]$ be an *upper bound* on the number of slots node i needs to transmit the aggregated data report to its parent for an instance of query q. If the planner constructs a plan for a query q, the same plan can be reused to execute a query q' if $W_q[i] \geq W_{q'}[i]$ for all nodes i. Examples of queries that share the same plan are the queries for the maximum temperature and the average humidity in a building. For both queries a node transmits one data report in a single step (i.e., $W_{max}[i] = W_{avg}[i] = 1$ for all nodes i) if the slot size is sufficiently large to hold two values. For the max query, the outgoing packet includes the maximum value of the data reports from itself and its children. For the average query, the packet includes the sum of the values and the number of data sources that contributed to the sum.

DCQS amortizes the overhead of computing query plans by executing multiple queries according to the same plan. This is often possible since queries with different temporal properties may be executed according to the same plan. We say that two queries belong to the same *query class* if they may be executed according to the same plan.

The precision to which the workload demand can be computed depends on the nature of the aggregation function. Three cases are worth highlighting. First, when statistical functions such as min, max, average, or histograms are computed over a set of sensors, the workload on each node remains constant, as seen in the prior examples. Second, when sensors produce data at a constant rate, the load of each node may be computed easily based on its location in the routing tree by summing the workload of the descendant nodes and that of the node and dividing it by the size of a packet. For these two common uses, the workload of each node may be computed precisely. In the case when a sensor produces data at variable rate, we advocate for constructing plans for the maximum data rate produced by each source. While this results in some internal fragmentation when the actual rate is below the maximum data rate, it keeps the number of plans to a minimum.

It is worth highlighting that DCQS supports the case when queries involve overlapping node subsets. In this case, DCQS would construct a plan for each query and compute the minimum inter-release time between the two plans. Nodes shared by multiple

queries will have higher transmission demand including transmissions for all queries in which they are involved. In addition, tt is important to note that DCQS does not construct the optimal schedule for executing but rather constructs the plans for each query in isolation. While this may result in suboptimal solutions, the construction of independent plans executed concurrently by enforcing the minimum step distance enables the development schedulers with small run-time overhead which are essential for dynamically determining which steps may be executed without conflict at run-time.

4.2.3 The Scheduler

In this subsection, we first describe how to construct a global conflict-free schedule. We then present an efficient local scheduling algorithm. For clarity, in this subsection we assume that all queries are executed according to the same plan, i.e., they belong to the same query class. We extend our solution to handle multiple query classes in the next subsection.

Definitions. Each instance is executed according to the plan of its query class. We use $E_{q,k}[s]$ to denote the set of transmissions assigned to step s of $I_{q,k}$'s plan. We say that two steps of instances $I_{q,k}$ and $I_{q',k'}$ are *conflict free* $E_{q,k}[s] \parallel E_{q',k'}[s']$ if all pairs of transmissions in $T_c[s] \cup T_{c'}[s']$ are conflict free. We also use the notation $E_{q,k}[s] \nparallel E_{q',k'}[s']$ to denote that the two steps conflict with each other. The scheduler executes steps such that: (1) All steps executed in a slot are conflict-free. (2) The relative order in which the steps of an instance are executed is preserved: if step $E_{q,k}[s]$ is executed in time slot i, step $E_{q,k}[s']$ is excuted in slot i' and $s > s'$ then $i > i'$. This ensures that the precedence constraints required by aggregation are enforced.

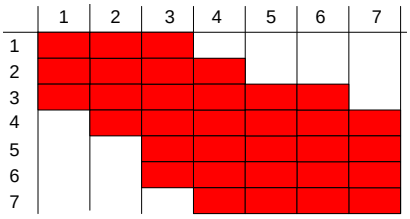
The Brute Force Approach. Let us consider a brute-force algorithm to dynamically determine what steps may be executed in a slot. We say step $E_{q,k}[s]$ is *ready* if $E_{q,k}[s-1]$ has been executed. The first step $E_{q,k}[1]$ is ready when the instance $I_{q,k}$ is released at time $R_{q,k}$. Intuitively, the brute force approach schedules in each slot multiple *conflict-free* and *ready* steps. Priority is given to executing steps in the transmissions plans of instances with earlier release times. To determine what steps may be executed in a slot, we need to know if any two steps in the plan conflict. To

facilitate this we construct a conflict table of size $L_q \times L_q$ that stores the conflicts between any pairs of steps in the plan of the query class. Figure 4.6(a) shows the conflict table of the plan presented in Figure 4.3. Figure 4.6(b) shows the transmission schedule constructed using the brute force approach under saturation conditions when an instance is released after the first step in the previous instance was executed.

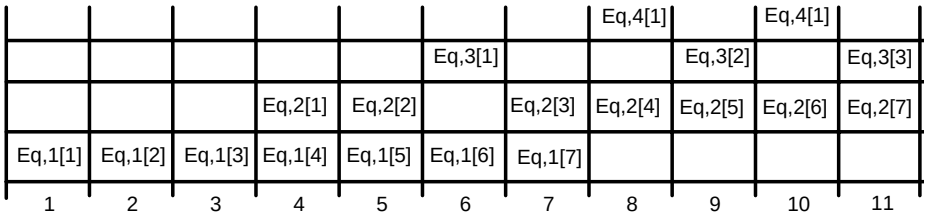
The brute force approach constructs the schedule as follows. Initially, $E_{q,1}[1]$ is the only step ready and it is executed in slot 1. In slot 2, the steps $E_{q,1}[2]$ and $E_{q,2}[1]$ are ready. However, the earliest slot when $E_{q,2}[1]$ may be executed is slot 4 since according to the conflict table $E_{q,2}[1] \nparallel E_{q,1}[1..3]$. So, in slot 4 we schedule $E_{q,1}[4]$ and $E_{q,2}[1]$. A more interesting case occurs when scheduling the steps in slot 6. In slot 6, $E_{q,1}[6]$ is executed since it has the earliest release time. $E_{q,2}[3]$ cannot be executed in slot 6 since $E_{q,2}[3] \nparallel E_{q,1}[6]$. However, $E_{q,3}[1]$ is ready and its execution does not conflict with $E_{q,1}[6]$. Therefore, it is also executed in slot 6. The process continues to construct the schedule presented in Figure 4.6(b).

The brute force approach is impractical due to its computation and storage costs. The computation time for determining what steps to schedule in a slot is quadratic in the number of ready steps in all instances that have been released. The memory cost for storing the conflict table is quadratic in the length of the plan. To alleviate these problems we may trade some of the throughput in favor of reduced computational and storage costs. To this end, we impose the additional constraint that the execution of an instance cannot be *preempted* (the reduction in throughput is characterized in Section 4.4). The execution of a query instance is not preempted if, once its first step is executed, the subsequent steps of its plan are executed *without gaps* in the following slots. For example, in Figure 4.6(b), the schedule constructed by the brute force approach does not meet this constraint since the execution of $I_{q,2}$ is preempted in slot 6.

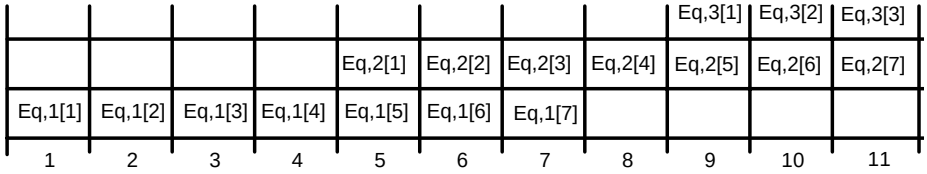
Minimum inter-release time. We define the *minimum inter-release time*, Δ, as the minimum number of slots the execution of $I_{q,k}$ must be delayed after another instance $I_{q',k'}$ starts executing such that the execution of $I_{q,k}$ and $I_{q',k'}$ are conflict-free. In other words, any two instances are conflict free as long as their inter-release time is at least Δ.



(a) Conflict table.



(b) Brute force approach.



(c) DCQS approach.

Figure 4.6: The conflict table captures the transmission conflicts between pairs of steps from the plan shown in Figure 4.4. The presence of a conflict is represented by the red rectangle. No rectangle indicates that the pair of steps may be executed concurrently. Based on the conflict table you can dynamically construct schedules either by brute force or through the DCQS approach.

Consider the execution of two consecutive instances $I_{q',k'}$ and $I_{q,k}$ (from one or two queries). If the inter-release time between $I_{q,k}$ and $I_{q',k'}$ is δ and the execution of instances cannot be preempted, then the steps $E_{q,k}[1]$ and $E_{q',k'}[\delta+1]$ are executed in the same slot of the transmission schedule. Hence, δ must be selected to ensure that $E_{q,k}[1] \parallel E_{q',k'}[\delta+1]$. However, the execution of $I_{q,k}$ may start in any slot after δ steps in the plan of $I_{q',k'}$ are executed. Therefore, we must guarantee that $E_{q,k}[1]$ does not conflict with $E_{q',k'}[\delta+1]$ *and* any of the subsequent slot executions i.e., $E_{q,k}[1] \parallel E_{q',k'}[\delta+i+1]$ for all $i \in [0, L_c-\delta-1]$, where L_c if the length of the plan of query class c. The minimum inter-release time, Δ, is the smallest number such that the execution of any step $E_{q,k}[s]$ does not conflict with $E_{q',k'}[s+\delta+i+1]$ where $s \leq L_c$ and $i \in [0, L_c-s-\delta-1]$. Thus, the minimum inter-release time is:

$$\Delta = \min_{\delta \in [1, L_c]}(E_{q,k}[s] \parallel E_{q',k'}[s + \delta + i + 1])$$
$$\forall i \in [0, L_c - s - \delta - 1], s \leq L_c \tag{4.1}$$

The minimum inter-release time is a measure of the degree of parallelism that may be achieved in query execution. In the worst case, when $\Delta = L_c$ a single instance may be executed at a time in the network.

The Scheduler. Each node employs a *local* scheduler that schedules the transmissions of all instances. The state maintained by the scheduler includes: the start time and period of all queries, the plan's length, and the minimum inter-release time. Note that as long as all nodes have a consistent view of these parameters, they will construct independently the same schedule. The scheduler also knows the steps in which the host node transmits or receives. However, the scheduler does not need to know the specific steps in which any other nodes transmit or receive.

The scheduler has two queues: a *run* queue and a *release* queue. Both the *run* and *release* queues are FIFO queues. The *release* queue contains all instances released but not being executed while the *run* contains the instances to be executed in slot s. Although the *run* queue may contain multiple instances, a node is involved in transmitting/receiving for at most one instance (otherwise, it would be involved in two conflicting operations). A node n determines if it transmits/receives in slot s by checking if it is assigned to transmit/receive in any of the steps to be executed in slot s. If a node does not transmit or receive in slot s, it turns off its radio for the duration of the slot.

The scheduler enforces a minimum inter-release time of at least Δ between the start time of any two instances by starting an instance in two cases: (1) when no instances are executed (i.e., *run*=$\emptyset$) and (2) when the step distance between the head of the *release* queue (i.e., the instance with the earlierst release time) and the tail of the *run* queue (i.e., the last instance that started) is larger Δ. When an instance starts, it is moved from the *release* queue to the *run* queue.

The scheduler is simple and efficient making it feasible to run it on resource-constrained devices. When a new instance is released, the scheduler inserts it in the *release* queue. In each slot, DCQS determines what instances should start. This operation takes $O(1)$, since it involves comparing the step distance between the instances at the head of *release* queue and tail of *run* queue with the minimum step distance. To determine if a node should send, receive, or sleep, DCQS iterates through the instances in the *run* queue. This requires $O(|run|)$ time if each node maintains a bit vector indicating whether it transmits, receives, or sleeps in each step of a plan. Thus, the complexity of the operations performed in a slot is $O(|run|)$. Due to the efficiency of the scheduler, a node may construct the transmission schedule dynamically at run-time based on the properties of admitted queries, plan length and minimum inter-release time. Second, the memory cost of the algorithm is also significantly lower than the brute force approach. The scheduler maintains only the minimum inter-release time instead of a table of conflicts as in the brute-force approach.

Figure 4.6(c) presents the schedule constructed when the minimum inter-release time Δ is 4 slots. The constructed schedule has slightly lower throughput than the one constructed using the brute force approach. This is due to the fact that DCQS does not preempt instances once they are executing. This illustrates our decision to trade-off throughput to reduce the memory and processing costs. However, our simulation results show that DCQS still achieves significantly higher throughput than existing solutions (see Section 4.4).

Analysis. In the following we prove three properties of the DCQS scheduler. First, we prove that the scheduler never schedules conflicting transmissions in the same slot. Second, we analyze the network capacity in terms of query completion rate under DCQS. This result is important because it enables us to prevent the network workload to exceed DCQS's capacity (as described in Section 4.3.2). Finally, we characterize the energy consumption of a node.

Theorem 1 *The scheduler executes conflict-free transmissions in all slots.*

Proof 1 *Consider the scheduler constructing a schedule for the following sequence of instances* $I_{q_1,k_1}, I_{q_2,k_2}, I_{q_3,k_3} \cdots$. *We note that the scheduler ensures that the pairs* I_{q_1,k_1}, I_{q_2,k_2} *and* I_{q_2,k_2}, I_{q_3,k_3} *are conflict-free, but it does not directly ensures that* I_{q_1,k_1}

and I_{q_3,k_3} are conflict free. In general, we must prove that I_{q_i,k_i} does not result in any conflict when its schedule overlaps with any instance I_{q_j,k_j} where $j > i$.

Let s_i and s_j be the steps in the plans of I_{q_i,k_i} and I_{q_j,k_j} that the scheduler assigns in the same slot. Since the scheduler enforces a minimum inter-release time of Δ between consecutive instances then $s_j - s_i \geq (j-i) \cdot \Delta \geq \Delta$. Thus, the scheduler executes conflict-free transmission in any slot.

Theorem 2 *The maximum query rate of DCQS is $\frac{1}{\Delta \cdot slotSize}$ where slotSize is the size of a slot in seconds.*

Proof 2 *An instance can be released every Δ slots for a maximum query completion rate of $\frac{1}{\Delta \cdot slotSize}$. A consequence, DCQS does not exceed its capacity if:*

$$\sum_{q} \frac{\Delta}{P_q / slotSize} \leq 1 \tag{4.2}$$

where, P_q is the period of query q.

A network running DCQS has predictable power consumption. DCQS keeps a node n awake only when it or one of its children is scheduled to transmit a data report. Otherwise, node n is scheduled to sleep. Therefore, the power consumed by n to execute a query q is:

$$Pwr_n(q) = \frac{1}{P_q} \cdot (Pwr_{recv} \cdot \sum_{c \in child(n)} W_q[c] + Pwr_{send} \cdot W_q[n]) \tag{4.3}$$

The rate of query q is $\frac{1}{P_q}$. $W_q[c]$ is the maximum number of packets a child c transmits to n to satisfy the workload demand of q. $W_q[n]$ is the maximum number of packets transmitted by n to its parent. Pwr_{recv} and Pwr_{send} is the power consumed in receiving and transmitting a packet, respectively. Based on Equation 4.3 the network lifetime may be computed.

4.2.4 The Multiple Query Class Scheduler

We now extend our scheduler to support multiple query classes. To this end, we must refine the definition of minimum inter-release time to accommodate the case when instances are executed according to different plans. We define the *minimum inter-release of query classes c and c'* $\Delta(c, c')$ as the minimum number of slots an instance of class c' must wait after an instance of class c started such that there are no conflicts. Note that Δ is not commutative.

Given the minimum inter-release times between any ordered pairs of query classes, the scheduler needs to control the inter-release times of two consecutive query instances based on their query classes. We note that the storage cost of the multiple class scheduler is quadratic in the number of query classes, since it must store the minimum inter-release time of each pair of query classes. However, as discussed in Section 4.2.2, the number of query classes in a WSN is usually small.

When all queries belong to a single query class, the scheduler only needs to check if the step difference between the instance at the head of the *release* queue and the instance at the tail of the *run* queue exceeds the minimum inter-release time to guarantee conflict-free transmissions. However, in the case of multiple query classes, to guaranteee that *all* minimum inter-release times are enforced, the scheduler should check if the step difference between the instance at the head of *release* queue and *all* instances in *run* queue exceeds the minimum inter-release times between their respective query classes. An efficient mechanism for doing this is for the scheduler to keep track of the slot when the last instance of each query class started. To enforce *all* minimum inter-release times it suffices for the time when the last instance of each class started to exceed the minimum inter-release time between that class and the class of the instance at the head of the *release* queue. Thus, the number of comparisons necessary to enforce the minimum inter-release time equals the number of query classes. As a consequence, the scheduler handles multiple classes without increasing its computational complexity significantly since the number of classes is a constant (i.e., it does not depend on the number of instances either in *release* or in *run* queues).

Equation 4.2 allows us to compute DCQS's maximum query throughput for a single query class. It is easy to see that a conservative bound on the maximum query rate for multiple classes is at least $\frac{1}{\Delta_{max} \cdot slotSize}$, where Δ_{max} is the maximum minimum inter-release time for all pairs of query classes. However, this approach significantly underestimates the maximum query rate supported by DCQS particularly when the values of Δ differ significantly. To reduce the pessimism of the bound, we now derive a *sufficient* condition for ensuring that all queries may be scheduled without exceeding the network capacity:

Theorem 3 *Given a set of queries classes C ($|C| \geq 1$), all queries can be scheduled by DCQS without exceeding network capacity if:*

$$\sum_{q} \frac{\max_{c' \in C} \Delta(cls(q), c')}{P_q / slotSize} \leq 1 \tag{4.4}$$

where $cls(q)$ is q's query class, , P_q is q's period, and $slotSize$ is size of slot in seconds.

Proof 3 *Consider a query instance $I_{q,k}$ of class $c = cls(q)$. Any query instance $I_{q',k'}$ of class c' may start after $I_{q,k}$ completes $\Delta(c, c')$ steps in its execution. As such, in the worst case, $I_{q,k}$ will delay the execution of any query instance $I_{q',k'}$ for at most $\max_{c' \in C} \Delta(c, c')$. In other words, $I_{q,k}$ prevents other query instances from being executed for at most $\max_{c' \in C} \Delta(c, c')$. Hence, the network utilization of q, i.e., fraction of time the network executes q alone is: $\frac{\max_{c' \in C} \Delta(c,c')}{P_q / slotSize}$. Thus, DCQS does not exceed its capacity if there is sufficient time to execute all queries, i.e., the total utilization of all queries does not exceed 1:*

$$\sum_{q} \frac{\max_{c' \in C} \Delta(cls(q), c')}{P_q / slotSize} \leq 1$$

While (4.4) is still a conservative bound on query capacity, as shown in our simulation study presented in Section 4.4, it is close to the actual achievable throughput and hence is suitable for online rate control.

4.2.5 Distributed Planner

In this subsection we present a distributed planner which uses only neighborhood information in constructing plans. Specifically, a node knows only its adjacent communication and interference edges. We say that a node is in n's *one-hop neighborhood* if there is a communication or interference edge between it and n. The two hop neighborhood of node n includes n's one-hop neighbors and their one-hop neighbors. After running the decentralized planner a node knows its *local plan* which contains the step assignments for its two-hop neighbors.

To construct a local plan, a node communicates only with its one-hop neighbors. However, some of the neighbors may lie outside the node's communication range. A routing algorithm or limited flooding may be used to communicate with these nodes over multiple hops.

A node n constructs a plan in three stages: plan formulation, plan dissemination, and plan reversal. The formulation stage starts when a node n becomes the highest-priority eligible node in its one-hop neighborhood. When this occurs, n broadcasts a *Plan Request* packet to gather information about transmissions which have already been assigned steps. To construct a conflict-free plan, n must know the steps in which its two-hop neighbors with higher priorities were assigned. Upon receiving the *Plan Request* from n, each one-hop neighbor checks if there is a node in its own one-hop neighborhood that has a higher priority than n. If no such node exists, the receiver responds with a *Plan Feedback* packet containing its local plan. Otherwise, the node does not reply. After a time-out, node n will retransmit the *Plan Request* to get any missing *Plan Feedback* from its one-hop neighbors. Since all *Plan Feedback* are destined for n, to reduce the probability of packet collisions, nodes randomize their transmissions in a small window. Once n receives the *Plan Feedback*, it has sufficient information to assign its transmissions to its parent using the same method as the centralized planner. In the second stage, n disseminates its local plan to its one-hop neighbors via a *Plan Send* packet. Upon receiving a *Plan Send*, a node updates its plan and acknowledges its action via a *Plan Commit* message.

To ensure that DCQS constructs a conflict-free schedule, neighboring nodes must have consistent plans. We note that the distributed planner achieves this objective through

retransmission when needed. If a *Plan Feedback* message from some neighbors are lost, node n assumes that a higher priority node has not yet been scheduled and retransmits the *Plan Request* until it has received Plan Feedback from each neighbor or reached the maximum number of re-transmissions. Similarly, during the plan dissemination stage, node n retransmits the plan until all its neighbors acknowledge the correct reception of its *Plan Send* via the *Plan Commit* message.

Finally, the planner reverses the plan. To do this, a node must know the length of the global plan. We take advantage of the routing tree and data aggregation to compute the length of the plan as follows. A node computes the length of its local transmission plan length based on the maximum step number in which a transmission/reception is assigned. The node aggregates its local length of the plan with that of its children by taking the maximum of the two. The result is sent to its parent. At the base-station, the plan length may be computed. The root then uses the routing tree to disseminate this value to each node. Upon receiving the plan length a node reverses its plans.

It is important to note that DCQS relies on nodes having consistent state for proper execution of plans. We ensure that this is the case by bounding the time for each phase of the plan construction. If the planning process fails DCQS will abort the construction of the plan if the plan construction does not succeed within the allotted time. The node that failed to respond during plan construction is considered disconnected and removed from the IC graph. At this point the process of plan construction is restarted. There is no point at which DCQS will start using a plan, before all nodes are synchronized with respect to the plan they are using.

Distributed computation of minimum inter-release times.

We now enhance the distributed planner to compute the minimum inter-release times.The key to computing the minimum inter-release time in a distributed manner lies in the observation that a node may compute its local value for the minimum inter-release time based on the its local plan and its local knowledge of the IC graph according to (4.1). The minimum inter-release time of the global plan is the maximum of the minimum inter-release times of the local plans. This suggests that, similar to the length of the plan, the global minimum inter-release time can be computed using in-network aggregation. In fact, the two may be computed concurrently. Once

the aggregation process is complete, the root can compute the length, and minimum inter-release time of the plan and then disseminate them to all nodes in the network.

4.3 Handling Dynamics

4.3.1 Dynamic Workload

DCQS can efficiently adapt to changes in the workload including arrival, deletion, and rate change of queries. Consider the case where a user issues a new query. The query service disseminates the query type and parameters to all nodes in the network. Next, DCQS checks if a transmission plan for the issued query was previously constructed. If no transmission plan was constructed, DCQS uses the decentralized planner to compute a new transmission plan and the minimum inter-release times. DCQS isolates the execution of current queries from the setup of new queries by periodically reserving slots for protocol maintenance. During the protocol maintenance slots, the planner computes the transmission plan and minimum inter-release times. Once they are computed, the scheduler has sufficient information to construct a conflict-free schedule which accounts for execution of the new query.

If a query from the same class was previously issued, a transmission plan for that class has already been constructed. As previously mentioned, queries from the same class share the same transmission plans and minimum inter-release time. Since usually there are only a small number of query classes, it is likely that DCQS already computed the transmission plan and minimum inter-release times of a class. In this case, DCQS can handle the new query without any additional overhead. Similarly, DCQS can also handle query deletions and rate changes of existing queries without any overhead.

4.3.2 Preventing Overload

A key advantage of DCQS is that it has a known capacity bound in terms of the maximum query completion rate. Using Equation 4.4, we can easily detect overload

conditions which obviates the need for complex congestion control mechanisms. When the user issues a query, DCQS uses the inequality in Equation 4.4 to determine if the capacity is exceeded. If the capacity is exceeded, we consider the following two options. First, the user may be notified that the query will not be executed because the network capacity would be exceeded. Second, DCQS may reduce the rates of existing queries to allow the new query to be executed. For example, a simple rate control policy is to reduce the rates of all queries proportionally by multiplying their rates by $\alpha = \left(\sum_q \frac{\max_{c' \in C} \Delta(cls(q),c')}{P_q/slotSize}\right)^{-1}$. This rate control policy is used in our simulations. As discussed in the previous section, DCQS may modify the period of a query without recomputing the transmission plan or minimum inter-release times. Therefore, the only overhead is to disseminate the new rates of the existing queries to the network.

4.3.3 Robustness Against Network Changes

We now describe how DCQS handles topology changes due to node or link failures. For DCQS to detect topology changes, we increase the slot size to allow a parent to acknowledge the correct reception of a data report from its child. A child can detect the failure of its parent or their link if it does not receive ACKs from its parent for several consecutive transmissions. A parent detects a child failure if it does not receive any data reports from that child for a number of query periods.

For all nodes to maintain a consistent schedule, DCQS must ensure the following: (1) the *two-hop neighbors* of a node have a consistent view of its local transmission plan, which dictates when the node transmits and receives data reports; (2) *all nodes* have consistent information about the length of the transmission plans and the minimum inter-release times. In response to topology changes, the routing tree must be adjusted. Consider the case when a node n detects that its parent p failed and, as a result, it must select a new parent p'. This entails the planner assigning a step in the transmission plan for $\overrightarrow{np'}$, while the step in which the transmission $\overrightarrow{np}$ is scheduled must be reclaimed. If $\overrightarrow{np'}$ can be assigned to the step in which $\overrightarrow{np}$ was scheduled or a different step *without conflicts* then DCQS only needs to update the local transmission plan. This involves node n disseminating its updated transmission plan to its two-hop neighbors. If this is not possible, then DCQS must start recomputing a

new transmission plan. We note that the computation of the new plan affects only nodes with lower priority than n. If during the computation of the plan either the minimum inter-release times or the transmission plan lengths are modified, this information must be disseminated to all nodes in the network. Consider the case when a child node c of n failed. In this case, the step in which c is assigned should be reclaimed. To reduce the overhead, DCQS reclaims such slots only when other topology changes occur.

To reduce the cost of handling topology changes, we now describe an approach to constructing robust transmission plans that can tolerate some topology changes. To handle this we change the mechanism used to adapt the routing tree in response to link or node failure. We allow a node to change its parent in the routing tree as long as the new parent is selected from a predefined *set of potential parents*. Our goal is to construct transmission plans that are insensitive to a node changing its parent under the constraint that the new parent is in the set of potential parents. To this end, we introduce the concept of *virtual transmissions*. Although node n actually transmits to a single potential parent, we construct the transmission plan and compute the minimum inter-release times as if n transmits to *all* potential parents. We trade-off some of the throughput in favor of better tolerating topology changes. This trade-off is similar to other TDMA algorithms designed to handle topology changes [29][63].

4.3.4 Robustness Against Variations in Link Quality

Wireless links are known to have variable quality as environmental conditions change. During the computation of workload demands for each node, the user must allocate sufficient time slots for potential packet retransmission to ensure reliability. DCQS already provides some robustness against change in link quality by having multiple parents among which a node may switch. However, DCQS may also account for variations in link quality through a different mechanism. DCQS can accommodate such changes by increasing the workload of a link based on its quality. For example, link layer commonly computes the expected number of transmissions (ETX) to necessary for correctly delivering a packet. DCQS could allocate a node to transmit up to ETX times a packet to ensure reliable delivery. We note that to ensure robustness, a parent

is forced to transmit at the scheduled time even if it did not receive all data reports from its children.

4.3.5 Supporting Other Traffic

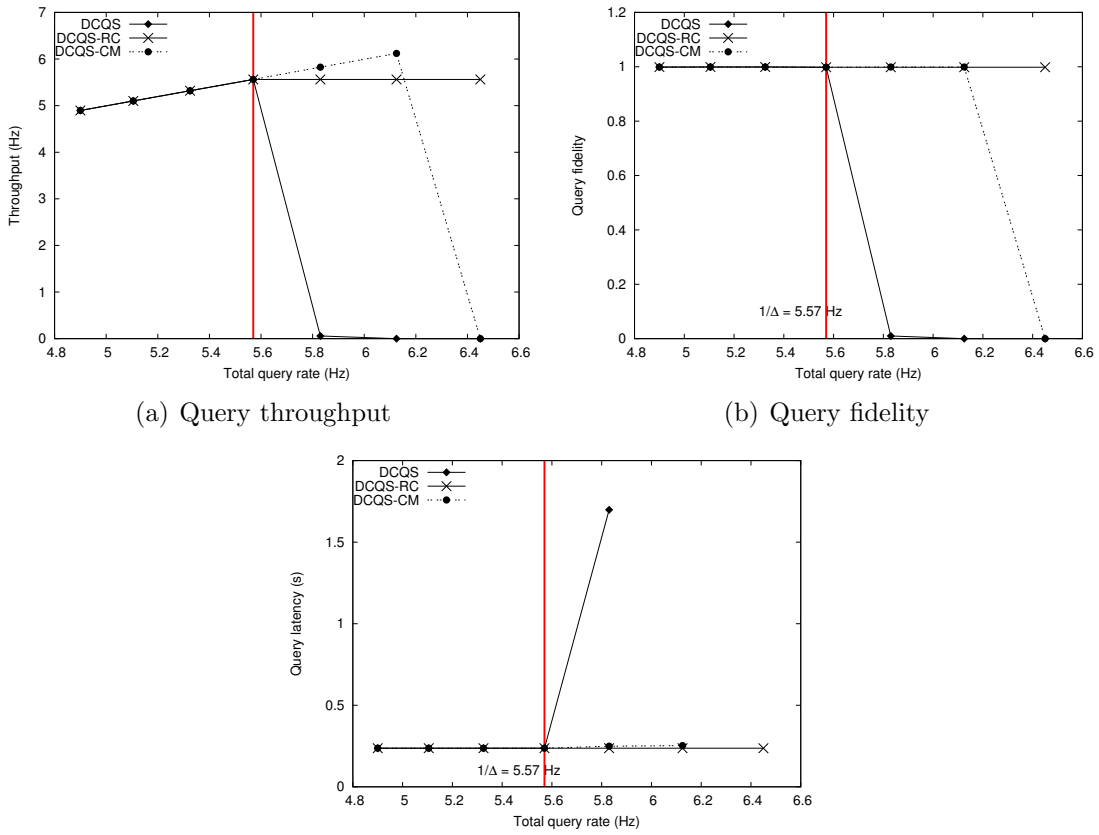
DCQS is optimized for improving the performance of periodic queries. However, other types of traffic may also exist (e.g., data dissemination, aperiodic queries). The simplest solution for handling these transmissions is to periodically dedicate slots for their transmission. Transmissions during these slots are done using typical CSMA/CA techniques. This approach reserves a portion of the bandwidth for other types of traffic. The proposed approach has the advantage of guaranteeing that a portion of the bandwidth is dedicated for traffic other than periodic queries. Moreover, it is straightforward to account for these additional slots in our analysis.

4.3.6 Time Synchronization

DCQS requires that all nodes within the interference range be time synchronized. The sensor network community has proposed several efficient time synchronization protocols [96]. For example, FTSP has an average time synchronization error of 1.4 μs per hop with minimal communication overhead. To account for time synchronization errors, DCQS employs guard-time intervals. Accordingly, in the beginning as slot, a short guard-time interval is observed by all nodes to account for clock drift. As shown in earlier publications, the use of guard time alleviates the need for fine-grained clock synchronization [106].

4.4 Performance Evaluation

We implemented the distributed version of DCQS in NS2. We used a two-ray propagation model at the physical layer. Since DCQS is targeted at *high* data rate applications such as structural health monitoring we configured our simulator according to the 802.11b settings. An overview of those deployments can be found in [88].



(a) Query throughput

(b) Query fidelity

(c) Query latency

Figure 4.7: Validation of capacity bound for single query.

In our simulations, the network bandwidth is 2Mbps. The communication range is 125m. The power consumed for transmitting and receiving a packet is 1.6W and 1.4W, respectively[8]. Each simulation run takes 200s.

In the beginning of the simulation, the routing tree is constructed. The node closest to the center of the topology is selected as the base station. The base station initiates the construction of the routing tree by flooding setup requests. A node may receive multiple setup requests from different nodes. The node selects as its parent the node that has the highest RSS among those with smaller depth than itself. The interference edges in the IC graph are determined using the method described in Section 4.1.2.

[8]Note that DCQS turns off the wireless interface whenever it is neither transmitting nor receiving.

We determined the slot size in the context structural health monitoring application as follows: we assume that a node samples its accelerometer at 100Hz and buffers 50 16-bit data points before transmitting its data report to its parent. To reduce the number of transmissions, data merging is employed: a node waits to receive the data reports from its children and merges their readings with its own in a single data report which it sends to its parent. In our experiments, the maximum number of descendants of any node is 20, so the maximum size of a data report containing 16-bit measurements is 2KB. Accordingly, we set slot size to 8.16ms, which is large enough to transmit 2KB of data.

For performance comparison we ran two baselines: 802.11b[59] and DRAND[105]. 802.11b a is representative CSMA/CA-based protocol, while DRAND is state-of-the-art TDMA protocol. Unlike DCQS, DRAND does not account for the interference relationships among nodes. Hence, the schedule it constructs may still result in collisions. To avoid this problem, we modified DRAND to treat the interference edges in the IC graph as communication edges. We augmented DRAND with a sleep-scheduling policy: a node remains awake if DRAND schedules itself or one of its children to transmit; otherwise, the node is turned off.

We evaluate the performance of three versions of DCQS: DCQS, DCQS-RC, and DCQS-CM. DCQS and DCQS-RC denote DCQS running with or without rate control. In DCQS, the planner uses the minimum inter-release time to decide when different steps may be executed concurrently. As discussed in Section 4.2.3, by knowing the entire conflict matrix one may make better scheduling decisions. DCQS-CM uses the conflict matrix rather than the minimum inter-release time to make scheduling decisions. We will use DCQS-CM to evaluate the degree of pessimism introduced by the minimum inter-release time. Note that DCQS-CM uses significantly more memory and has longer run-time overhead than DCQS. As a result, DCQS-CM may not be suitable for memory or processing constraint devices.

Our performance evaluation focused on the following metrics: query throughput, query fidelity, and query latency. The query throughput is measured by the number of data reports delivered to the base station. During the simulations data reports may be dropped preventing some sources from contributing to the query result. We quantify the quality of a query result using the query fidelity metric. The query

fidelity is the ratio of the number of sources that contributed to the query result received by the base station and the total number of sources.

4.4.1 Single Query Class

The first set of experiments assumes that all queries belong to the same class. Under this setup we will (1) validate the analytical results on DCQS's capacity presented Section 4.3, (2) compare the performance of DCQS against the baselines when multiple queries are executed, (3) compare the scalability of DCQS to that of DRAND when the network size is varied, (4) evaluate DCQS's overhead in relation to the network size, and (5) evaluate the impact of the virtual transmissions on DCQS's performance.

Tightness of Capacity Bound

The first experiment is designed to validate our capacity result for the case of a single query and show the effectiveness of the rate control policy.

A single query is executed in the network. The results are obtained from a topology of size 675m×675m. The topology is divided into grids of 75m×75m. In each grid a node is placed at random. The presented data is from a single run. We chose to present results from a single run, because for different topologies DCQS constructs transmission plans with different Δ values. Under these settings, DCQS constructed a transmission plan with $\Delta = 22$ slots. According to Theorem 2 the maximum query rate that DCQS may support is $\frac{1}{22 \cdot 8.16\ ms} = 5.7\ Hz$. The vertical lines in Figure 4.7 indicates the DCQS's capacity. To validate the capacity bound the query rate is varied from 4.9 Hz to 6.45 Hz.

Figure 4.7(a) shows the number of delivered data reports as the query rate is increased. We observe that the increase in query rate is matched by an increase in the number of data reports received until DCQS's capacity (5.7 Hz) is reached. When workload exceeds the DCQS's capacity, the performance of DCQS degrades drastically. As discussed in Section 4.3.2 rate control may be used to avoid triggering the capacity

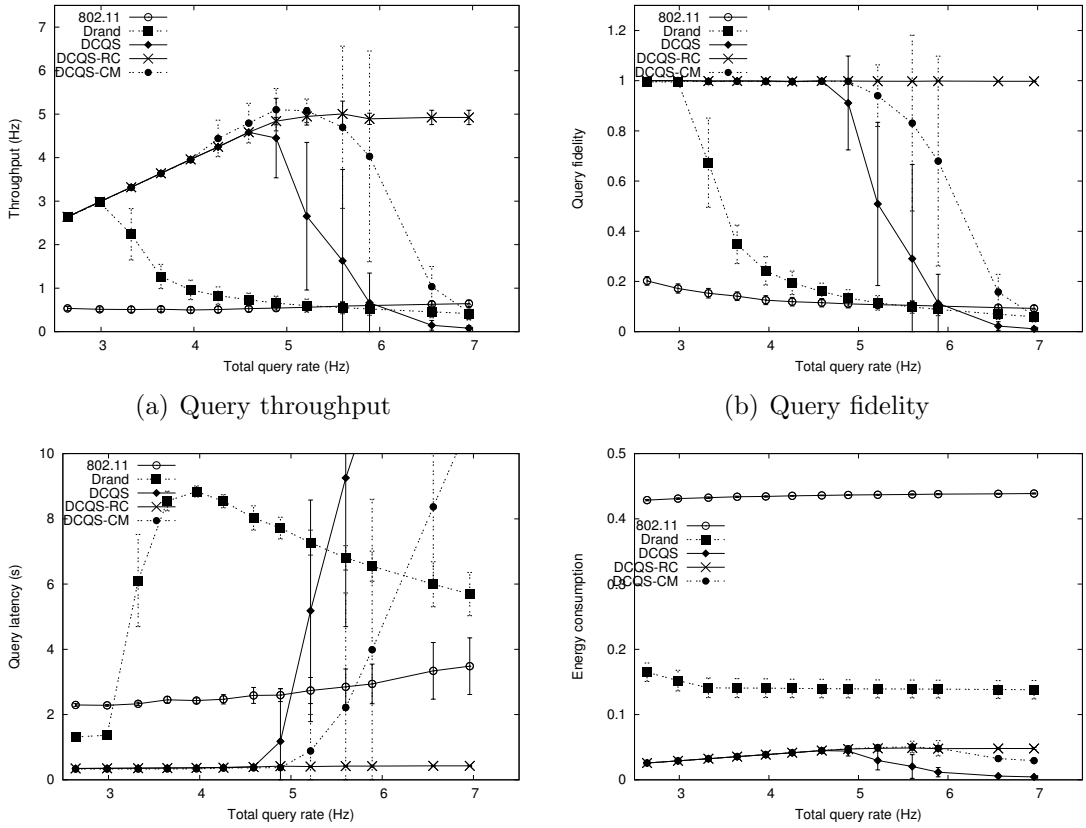
bottlenecks. As shown in the figure, DCQS-RC maintains its good performance even when the offered load exceeds the DCQS's capacity. Since DCQS-CM uses the conflict matrix, it may execute more concurrent steps than DCQS. As a result, DCQS-CM may support a query rate as high as 6.125 Hz, which is a 7.5% improvement of DCQS. However, this improvement is at the cost of increased memory and processing overheads.

Figure 4.7(b) shows the data fidelity. DCQS provides 100% query fidelity up its network capacity. This result shows that the schedule constructed by DCQS is conflict-free, validating the correctness of our algorithms. In addition, DCQS-RC, which uses the rate control policy, avoids the drop in query fidelity under overload conditions. A similar pattern may be observed in terms of delay as shown in Figure 4.7(c). DCQS, DCQS-RC, and DCQS-CM have similar latencies up to the DCQS's capacity. If the capacity is exceeded and the rate control is not used, the query latency increases sharply. In contrast, DCQS-RC is unaffected by the overload conditions.

Multiple Queries

This set of experiments is designed to compare the performance of DCQS to that of the baselines under different workloads. The workload is generated by running four queries with different rates. The ratio of the rates of the four queries $Q_1 : Q_2 : Q_3 : Q_4$ is 8:4:2:1. We refer to Q_1's as the base-rate. We vary the workload by changing the base rate. The start time of the queries is spread evenly in an interval of 81.6 ms. The topology setup is identical to the previous experiment. Each data point is the average of five runs. We also plot the 90% confidence interval for each point.

Figure 4.8(a) shows the query throughput as the total query rate is varied. A common trend may be observed: the protocols match the increase in the total query rate up to their respective maximum capacity and then their performance plummets. The lowest throughput is obtained by 802.11 protocol. The reason for this outcome is that the capacity of 802.11 is exceeded in all tested settings. This is because contention based protocols perform poorly under heavy workloads. DRAND outperforms 802.11 delivering all data up to a total query rate of 2.98 Hz. The DCQS variants all outperform DRAND. DCQS-RC achieves a maximum query rate of 5 HZ which is



(a) Query throughput

(b) Query fidelity

(c) Query latency

(d) Power consumption

Figure 4.8: Performance comparison when executing four queries and their base rate is varied.

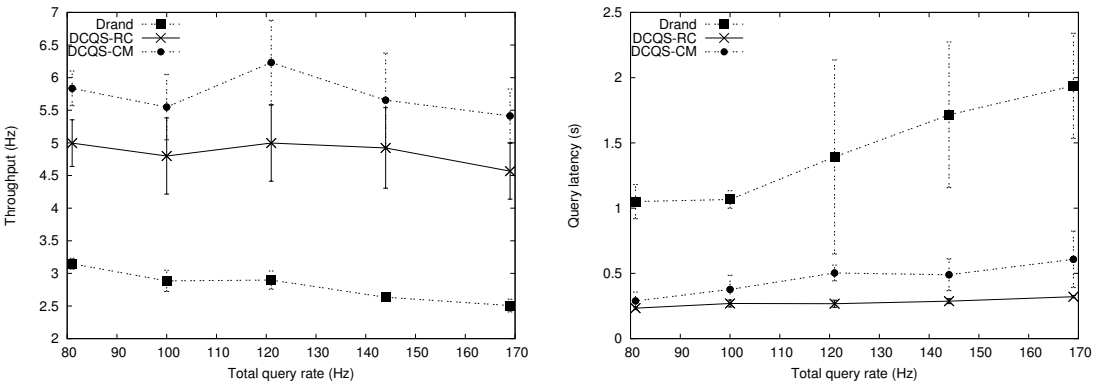
67% higher than DRAND. This result is attributed to the fact that DRAND assigns slots to nodes fairly. Fair allocation is unsuitable for queries in WSNs in which nodes have non-uniform workloads: for example, a node with more children need to receive more messages per each query instance. As in the previous experiment, DCQS-RC maintains its good performance even under overload conditions. This shows that our rate control policy works not only in the single-query case, but also in the multi-query case. DCQS-CM delivers all data reports up to a total query rate of 5.21 Hz which is an increase of 4% over DCQS-RC and 74% over DRAND. This shows that for the considered deployment, the reduction in throughput introduced by the minimum inter-release time is small.

Figure 4.8(b) shows the query fidelity of the protocols. As expected, 802.11 has poor query fidelity, whereas the TDMA protocols perform much better. DRAND maintains its high query fidelity up to its maximum query completion rate of 2.98Hz after which it plummets. The reason for this is that the transmission queues fill-up and packets are dropped. In contrast, DCQS-RC maintains 100% fidelity for all tested query rates.

Figure 4.8(c) shows the query latency of the presented protocols. Even when the query rate is low, DCQS has significantly better query latency than DRAND. For example, when the query rate is 2.64Hz, DRAND has a query latency of 1.31s. In contrast, DCQS has a latency of 0.35s which is 73% lower than that of DRAND. DRAND has a long query latency because at each hop a node may need to wait for the duration of an entire frame before it may transmit its packet to the parent. In contrast, DCQS accounts for the precedence constraints introduced by data aggregation when constructing the transmission plans. This results in a significant reduction in the query latency.

Figure 4.8(d) presents power consumed during the experiment. We observe a decrease in the power consumption by DRAND with the query rate up to 2.98Hz. The performance drastically degrades after this point due to packet loses. Even under light loads, DCQS performs better than DRAND in terms of power consumption. The reason is that DRAND must remain awake when a child is scheduled to transmit even if the child node has no packets to transmit. In contrast, DCQS takes advantage of temporal properties of the workload to wake-up nodes only when necessary. As observed in the previous set of experiments, The power consumed by DCQS increases linearly with the query rate.

This set of experiments indicates that DCQS significantly outperforms both 802.11 and DRAND in all the considered metrics. Two factors contribute DCQS's high performance. First, the planner constructs transmission plans based on a heuristic that accounts for the precedence constraints introduced by data aggregation. This is highly effective in reducing message latency. Second, the scheduler overlaps the execution of multiple query instances to increase the query completion rate.



(a) Query throughput (b) Query latency

Figure 4.9: Performance comparison for topologies of different sizes.

Impact of Topology Size

This is designed to evaluate the scalability of the TDMA protocols. To this end we constructed five topologies with an increasing number of nodes by increasing the deployment area and keeping the node density constant. All topologies are squares with edges of size 675m, 750m, 825m, and 900m. Each area is divided into grids of 75 m $\times$ 75 m. In each grid, a node is placed at random. Each data point is the average of five runs. The 90% confidence intervals are also plotted.

Figure 4.9(a) shows the number of data reports delivered by DCQS and DRAND for each topology. The maximum query completion rate of DCQS-RC was computed theoretically and then verified experimentally. To determine the maximum number of data reports which may be delivered by DRAND or DCQS-CM, we increased the query rate until the query fidelity dropped below 90%. This is reasonable since the DRAND and DCQS-CM drop packets only if a node's queue fills up. When the topology has 81 nodes, DCQS outperforms DRAND by 58%. When the topology contains 169 nodes, the performance gap between the two protocols increases, DCQS outperforming DRAND by 115%. The increasing performance gains of DCQS are the result of being able to pipeline the execution of queries as the networks become larger. While the difference between DCQS and DCQS-CM is between 16 – 24% without a clear trend, it is worth noting that 90% confidence intervals overlap indicating that the performance may not be statistically significant.

Figure 4.9(b) shows the query latency at the maximum query rate supported by each protocol. The query latency of all protocols DCQS increases with the topology size. However, DCQS's rate of increase is significantly lower than DRAND's. The key to understanding this result is that the one-hop delay of DRAND is significantly larger than that of DCQS. The one-hop delay corresponds to the slope of the shown curves. When using DRAND, a node often needs to wait for the entire length of a frame before it may transmit its packet. In contrast, DCQS has low one-hop delays. Two factors contribute to this. First, DCQS organizes its transmissions to account for the precedence constraints introduced by data aggregation. Second, DCQS executes multiple query instances concurrently. As such, the time until the query instance starts being executed is reduced. DCQS-CM has a slightly higher query latency than DCQS-RC. This can be justified by the fact that DCQS-CM buffers more queries for execution than DCQS-RC which reduces the query rates to avoid triggering the capacity bottlenecks.

Communication Costs

To evaluate the overhead of DCQS we have collected statistics regarding the different types of packets transmitted by DCQS. We distinguish the following categories. The tree construction category includes all packets exchanged during the construction of the routing tree. The ICG construction category includes all the packets exchanged for constructing the IC graph. The planer category includes all the overhead associated with constructing plans. Finally, the minimum inter-release category shows the overhead of computing the minimum inter-release time. The overhead in packets for the topologies considered in the previous experiment are shown in Figure 4.10.

Figure 4.10 indicates that the cost of constructing plans dominates the DCQS overhead. This highlights the importance of the plan sharing strategies. The planning overhead is linear in the number of nodes in the network. Note that we did not make any particular effort in optimizing the performance of the planner. In fact, the planner does not take advantage of the broadcast nature of the wireless medium using only unicast packets to disseminate the plans. Therefore, we expect that the scheduling cost of construct plans may be significantly reduced.

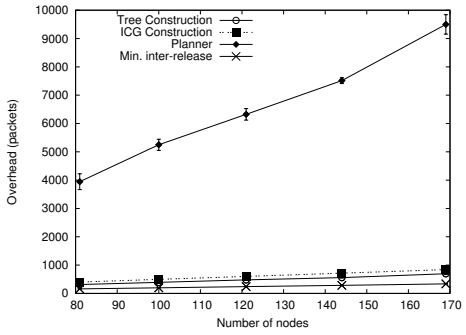
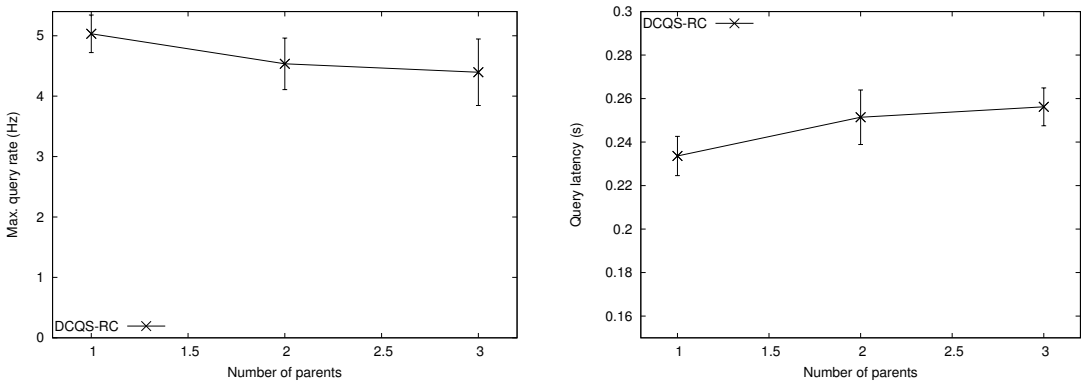


Figure 4.10: Communication costs for different size networks.



(a) Query throughput

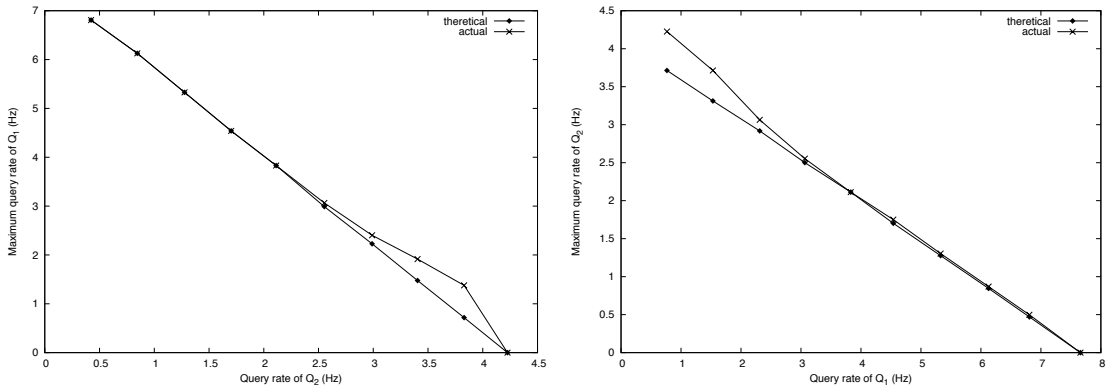
(b) Query latency

Figure 4.11: Impact of virtual transmissions on DCQS.

Virtual Transmissions

To evaluate the impact of the virtual transmissions approach presented in Section 4.3.3, we measured the performance of DCQS when the number of parents is varied from one to three. We measured the performance of DCQS-RC by the maximum query rate supported which was first computed theoretically and then verified empirically.In addition, we measured the query latency of DCQS-RC at the maximum query rate.

Figure 4.11(a) indicates that the DCQS's maximum query rate degrades as the number of potential parents is increased. The increase to having two or three potential parents leads in a throughput reduction of 9.8% and 12.6%, respectively. However, DCQS improved throughput over traditional TDMA protocols by as much as 58%. Therefore, even if DCQS would construct plans in which multiple potential parents, there still are significant performance gains.



(a) Q_1's tightness (b) Q_2's tightness

Figure 4.12: Validation the capacity bound for multiple query classes.

The use of virtual transmissions also increases the query latency as shown in Figure 4.11(b). The increase to two and three potential parents results in an increase in query latency of 7.6% and 9.6%, respectively. This is due to the fact that the addition of multiple potential parents introduces new precedence constraints that force a node to wait for a longer before receiving all data reports from its children.

4.4.2 Multiple Query Classes

The second set of experiments is used to evaluate DCQS with multiple query classes. In the first experiment we evaluate the tightness of our capacity bound. The next experiment compares the performance of DCQS to that of the baselines.

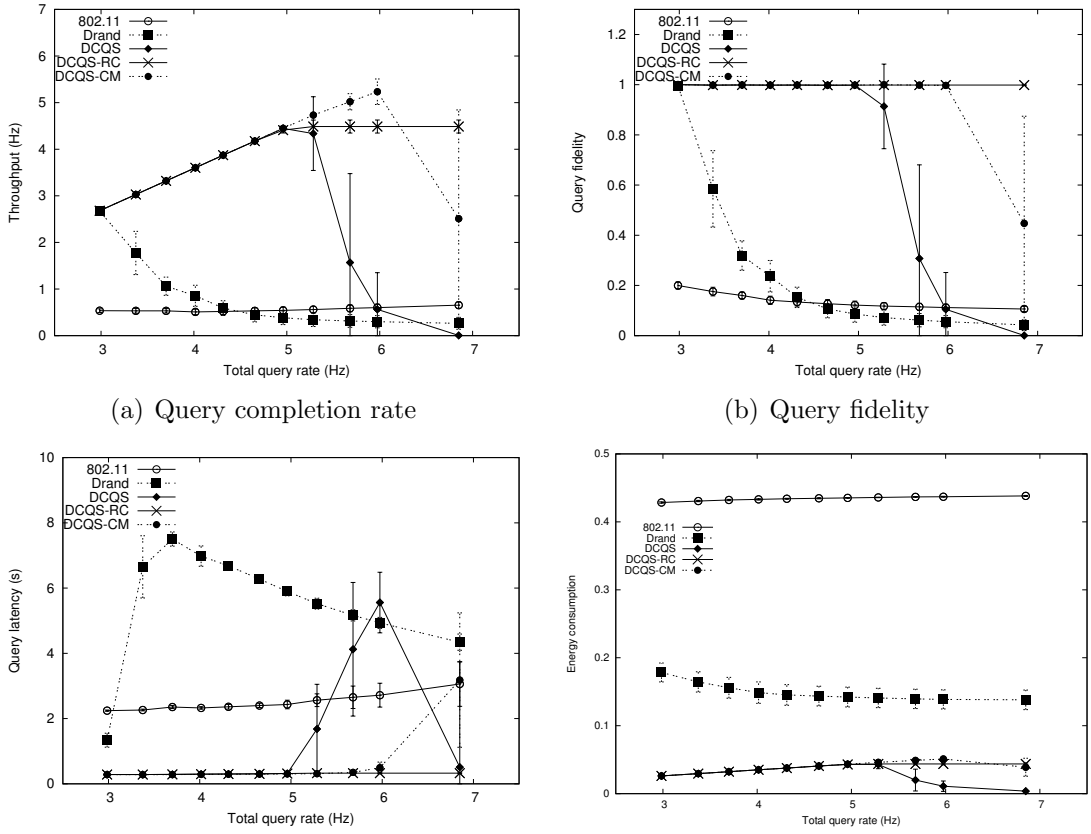
We create different query classes by varying the sources of the queries. For each query class we select at random a fraction of the leaf nodes as data sources. We note that if a node has as descendent a selected leaf node, then it also participates in that query class since it must forward the leaf's data to the base-station. Similar to the previous experiments, data merging is performed as data is routed to the base-station.

Tightness of Capacity Bound

This experiment uses the same network topology as the one presented in Section 4.4.1. However, different from the previous experiment, the workload includes two queries belonging to different query classes. The class of Q_1 involves 40% of the leaf nodes as sources, while the class of Q_2 involves all leaf nodes as sources. The obtained minimum inter-release times were $\Delta(c_1, c_1) = 16$ slots, $\Delta(c_1, c_2) = 14$ slots, $\Delta(c_2, c_1) = 29$ slots, and $\Delta(c_2, c_2) = 25$ slots.

In Figure 4.12(a) we fix the rate of query Q_2 and compare the theoretical maximum query rate of Q_1 against its actual maximum query rate achieved in the simulations. The *theoretical* maximum query rate of Q_1 is computed according to Equation 4.4. To evaluate the tightness of the capacity bound, we increase the rate Q_1 until packets start being dropped. We refer to the maximum query rate under which no packet is dropped as the *actual* maximum query rate. As shown in Figure 4.12(a) the theoretical maximum query rate never exceeds the actual maximum query rate. This result shows that the theoretical bound is a conservative bound of the actual query capacity. The theoretical maximum query rate remains the same when Q_2's rate is between 0.42 to 2.55Hz as shown in Figure 4.12(a). When Q_2's rate exceeds 2.55Hz, the discrepancy between Q_1's theoretical and actual maximum query rates increases. The maximum difference is 0.66Hz when Q_2's rate is 4.22Hz. The slight increased pessimism may be explained as follows. First, both the utilization of Q_1 and Q_2 is overestimated. For example, Q_2's utilization is overestimated because to enforce minimum inter-release time between two instances of Q_2 the scheduler uses $\Delta(c_2, c_2) = 25$; to enforce the minimum inter-release time between an instance of Q_2 followed by an instance of Q_1 the scheduler uses $\Delta(c_2, c_1) = 29$; in contrast, our capacity analysis uses the maximum of the two values to represent Q_2's utilization. A similar argument holds for Q_1. Second, since the sum of utilization of Q_1 and Q_2 must be smaller than 1, overestimating the utilization of Q_2 leads to underestimating the fraction of the capacity that may be used by Q_1, and the pessimism increases as the rate of Q_2 increases.

Similar to Figure 4.12(a), in Figure 4.12(b) we fix the query rate of Q_1 and plot theoretical and actual maximum query rates of Q_2. As in the previous experiment, Q_2's theoretical maximum query rate is slightly smaller than its actual maximum



(a) Query completion rate

(b) Query fidelity

(c) Query latency

(d) Power consumption

Figure 4.13: Performance comparison when three queries belonging to different query classes.

query rate validating the correctness of our capacity bound. The maximum difference between the Q_2's actual and predicted query rates is 0.51Hz.

Varying Query Rates

This set of experiments compares the performance of DCQS to that of the baselines under different query rates. The workload comprises three queries with different rates, each belonging to a different class. Q_1, Q_2, and Q_3 include 100%, 80%, and 60% of the leaf nodes as sources, respectively. The ratio of the rates of the three queries Q_1 :Q_2: Q_3 is 2.5:1.75:1. We refer to Q_1's rate as the base-rate. We vary the workload

by changing the base rate. Each data point is the average of five runs. The 90% confidence interval for each point is also plotted.

Figure 4.13(a) shows the query completion rate when the total query rate is varied. As expected, all protocols match the increase in the total query rate up to their respective capacity bounds after which the query completion rate degrades. 802.11b consistently has the lowest throughput. DRAND performs better than 802.11b achieving a maximum query completion rate of 2.66Hz. DCQS, DCQS-RC, and DCQS-CM outperform both baselines. DCQS-RC achieves a maximum query completion rate of 4.48Hz which represents a 62% improvement over DRAND. DCQS-RC avoids the performance degradation under overload conditions through rate control. DCQS-CM the highest query completion rate of 5.23 Hz, which is an improvement of 17% over DCQS-RC. Again, these performance benefits are the result of scheduling more steps concurrently and come at the cost of increase memory and processing costs.

Figure 4.13(b) shows the query fidelity as the total query rate are increased. As observed in the previous experiments, after a protocol exceeds its capacity, its query fidelity degrades drastically. DCQS-RC avoid the performance degradation caused by overload by using rate control. DCQS-RC achieves close to 100% fidelity in all experiments. This validates that DCQS schedules conflict-free transmissions even in the case of multiple query classes.This shows that our algorithms work correctly even in the presence of multiple query classes with involving overlapping subsets of nodes. This shows that our algorithms work correctly even in the presence of multiple query classes with involving overlapping subsets of nodes.

Figure 4.13(c) shows the average query latencies. When the total query rate is 2.98Hz, DCQS achieves a query latency of 0.28s compared to DRAND which had a query latency of 1.33s. This is reduction of over 78% in query latency. The significant reduction in latency highlights the of taking into account the precedence constraints between packet transmissions.

Figure 4.13(d) shows the power consumption of DCQS, DCQS-RC, and DRAND. DCQS and DCQS-RC consumed less power than DRAND. As expected, their power consumption increases linearly with the total query rate up to DCQS's capacity. After the DCQS's capacity is exceeded the power consumption of DCQS-RC remains

constant while DCQS's power consumption drops since fewer packets are delivered to the base-station.

4.5 Related Work

TDMA scheduling is attractive for high data rate sensor networks because it is energy efficient and may provide higher throughput than CSMA/CA protocols under heavy load. Two types of TDMA scheduling problems have been investigated in the literature: node scheduling and link scheduling. In node scheduling, the scheduler assigns slots to nodes whereas, in link scheduling, the scheduler assigns slots to links through which pairs of nodes communicate. In contrast to earlier work, DCQS adopts a novel approach which we call *query scheduling*. Instead of assigning slots to each node or link, we assign slots to transmissions based on the specific communication patterns and temporal properties of queries in WSNs. This allows DCQS to achieve high throughput and low latency.

Early TDMA scheduling protocols were designed for static or uniform workloads [31][37][103][7]. Such approaches are not suitable for dynamic applications with variable and non-uniform workloads. Several recent TDMA protocols can adapt to changes in workload. A common method to handle variable workloads is to have nodes periodically exchange traffic statistics and then adjust the TDMA schedule based on the observed workload [31][102][9]. However, exchanging traffic statistics frequently may introduce non-negligible communication overhead. In contrast, DCQS can efficiently adapt to changes in workloads by exploiting explicit query information provided by the query service. Furthermore, it features a local scheduling algorithm that can accommodate changes in query rates and additions/deletions of queries without explicitly reconstructing the schedule.

To combine the benefits of CSMA and TDMA, several hybrid protocols have been proposed [106][5]. For example, ZMAC [106] constructs a transmission schedule for each node but it allows nodes to contend for access to other nodes slots using channel polling. Similarly, Funneling-MAC [5], a hybrid protocol designed for data collection, constructs a TDMA schedule that involves only the nodes within a few hops of the sink (where high contention occurs) while the remainder of the nodes use a CSMA

protocol. As result, the overhead of maintaining a multi-hop TDMA schedule is reduced. As an efficient TDMA scheduling algorithm, DCQS may be integrated with either of the hybrid schemes proposed in [106] [5].

Recently, several theoretical bounds for wireless networks have been derived [49][77][3]. These bounds provided important insight on the fundamental limits of wireless networks. However, they cannot be directly applied to specific networks in practice because they are derived based on ideal assumptions. In contrast, in this chapter we derive a tight bound on the maximum query rate achieved under DCQS. Such a bound is of practical importance since it can be used to prevent network overload.

TinyDB [89] is a representative query service that allows a user to collect aggregated data from a sensor network through a routing tree. It employs a coarse-grained scheduling scheme that evenly divides the period of a query into communication slots for nodes at different levels in a routing tree. TinyDB does not address scheduling for multiple queries with different timing properties. Moreover, the schedule of each node is fixed and does not adapt to the workload. DCQS can be integrated with TinyDB to enhance its performance and flexibility.

4.6 Summary

This chapter presents DCQS, a novel transmission scheduling technique specifically designed for query services in wireless sensor networks. DCQS features a planner and a scheduler. The planner reduces query latency by constructing transmission plans based on the precedence constraints in in-network aggregation. The scheduler improves throughput by overlapping the transmissions of multiple query instances concurrently while enforcing a conflict-free schedule. Our simulation results show that DCQS achieves query completion rates at least 62% higher than DRAND, and query latencies at least 73% lower than DRAND. Furthermore, we derive the theoretical capacity bounds for DCQS that may be used to prevent network overhead through rate control.

Chapter 5

Real-Time Query Scheduling

Recent years have seen the emergence of wireless cyber-physical systems that must support real-time data collection at high data rates over wireless sensor networks (WSNs). Representative examples include emergency response [94], structural health monitoring [88], and process measurement and control [120]. Such systems pose significant challenges. First, the system must handle various types of traffic with different deadlines. For example, during an earthquake, the acceleration sensors mounted on a building must be sampled and their data delivered to the base station in a timely fashion to detect any structural damage. Such traffic should have higher priority than temperature data collected for climate control. Thus, a WSN protocol should provide *effective prioritization* between different network workloads generated by different applications while meeting their respective deadlines. Second, the system must support *high throughput* since sensors may generate high volumes of traffic. For example, structural health monitoring require a large number of acceleration sensors to be sampled at high rates generating high network loads [135][69]. Furthermore, since the system must deliver data to base stations or users within their deadlines, it is important for the system to achieve *predictable and bounded end-to-end latencies.*

Many wireless cyber-physical systems use query services to periodically collect data from sensors to a base station over a multi-hop wireless network. In this chapter, we propose ***R****eal-****T****ime* ***Q****uery* ***S****cheduling* (RTQS), a transmission scheduling approach for real-time queries in WSNs. To meet this challenge, we present three new real-time query scheduling algorithms and associated schedulability analysis. Our scheduling algorithms exploit the unique characteristics of queries in WSN including many-to-one communication and periodic timing properties. Furthermore, we derive the upper

bounds of end-to-end latencies of real-time queries. A unique aspect of our analysis approach is that it *bridges the gap between wireless sensor networks and schedulability analysis techniques which have traditionally been applied to real-time processor scheduling.*

This chapter makes four contributions: First, we show through analysis and experiments that query scheduling has an inherent tradeoff between prioritization and throughput. Second, we developed three scheduling algorithms: (1) The *nonpreemptive* query scheduling algorithm achieves high throughput at the cost of some priority inversions. (2) The *preemptive* query scheduling algorithm eliminates priority inversion at the cost of reduced throughput. (3) The *slack stealing* scheduling algorithm combines the advantages of preemptive and non-preemptive scheduling algorithms by improving the throughput while meeting query deadlines. Third, we provide schedulability analysis for each scheduling algorithm. Our analysis enables predictable real-time query services through online admission and rate control. Finally, we provide simulation results that demonstrate the advantages of RTQS over contention-based and TDMA-based protocols in terms of both real-time performance and throughput.

The chapter is organized as follows. Section 5.1 compares our approach to existing work. Section 5.2 details the design and analysis of RTQS which are later extended in Section 5.3 to the case when there are multiple query classes. Section 5.5 provides simulation results. Section 5.6 concludes the chapter.

5.1 Related Work

Real-time communication protocols may adopt contention-based or TDMA-based approaches. Contention-based protocols support real-time communication through probabilistic differentiation. This is usually achieved by adapting the parameters of the CSMA/CA mechanism such as the contention window and/or initial back-off [46][2][64][95]. Overviews of these approaches are presented in [146] and [98]. Rate and admission control [65][4][10][137] have also been proposed for contention-based protocols to handle overload conditions. However, contention-based approaches have two inherent drawbacks that make them unsuitable for high data rate and real-time

applications. First, communication latencies are highly variable due to the random back-off mechanisms. Second, their throughput drops significantly under heavy load due to excessive channel contention.

The approaches discussed so far only consider implementing real-time communication at the MAC layer. Several prior works argue that better real-time performance may be achieved in multi-hop networks if real-time protocols are implemented at the routing layer [51][40][25]. For example, the SPEED protocol [51] builds upon geographic routing to deliver packets at an uniform velocity in a multi-hop network. Extensions to the SPEED protocol were proposed to support multiple delivery speeds through multi-path routing [40] or power control [25]. These protocols commonly build upon contention based MAC protocols and, as a result, inherit their drawbacks.

TDMA protocols can achieve predictable communication latencies and higher throughput than contention-based protocols under heavy load. Several real-time TDMA protocols were designed for single-hop networks. The IEEE 802.15.4 standard specifies Guaranteed Time Slots (GTS) for achieving predictable delays in single hop networks. A flexible slot reservation mechanism was proposed [73] where slots are allocated based on delay or bandwidth requirements.

Several protocols aim at supporting real-time communication in multi-hop networks. Two recent chapters proposed real-time transmission scheduling for robots [38][81]. Both protocols assume that at least one robot has complete knowledge of the robots' positions and/or network topology. While the protocols may work well for small teams of robots, they are not suitable for queries in large-scale WSNs. Implicit EDF [17] provides prioritization in a single-hop cell. The protocol supports multi-hop communication by assigning different frequencies to cells with potential conflicts. However, the protocol does not provide prioritization for transmitting packets across cells. In contrast, RTQS provides prioritization in multi-hop networks and does not require multiple frequencies.

In [3] the real-time capacity of a network is characterized providing important insight on the fundamental limits of real-time communication in wireless networks. However, the results cannot be applied in practice because they are derived under ideal assumptions. In contrast, this chapter provides schedulability analysis for determining if queries may meet their deadlines.

Two recent protocols were designed support real-time flows. In [94] a scheduling based solution is proposed to support voice streaming over real-time flows. The real-time chains protocol [15] extends a contention-based scheme called Black Burst to support packet prioritization. However, these protocols only support real-time *flows* involving only one or a few data sources. In contrast, RTQS is optimized for real-time *queries* that collect sensor data from many sources.

In previous chapter we described DCQS, a transmission scheduling technique for WSN queries. In contrast to traditional TDMA protocols designed to support general workloads, DCQS is specifically designed to exploit specific communication patterns and temporal properties of queries in WSNs. This allows DCQS to achieve high throughput and low latency. However, DCQS does not support query prioritization or real-time communication, which is the focus of this chapter.

5.2 Real-time Query Scheduling

RTQS achieves predictable and differentiated query latencies through prioritized conflict-free transmission scheduling. Our approach relies on two components: a *planner* and a *scheduler*. The planner constructs a *plan* for executing all the instances of a query. A plan is an ordered sequence of *steps*, each comprised of a set of conflict-free transmissions. RTQS employs the same distributed algorithm as DCQS to construct plans. The scheduler that runs on every node determines the time slot in which each step in a plan is executed. To improve the throughput, the scheduler may execute steps from multiple query instances in the same slot as long as they do not conflict with each other.

RTQS works as follows: (1) When a query is submitted, RTQS identifies a plan for its execution. As discussed in Section 5.2.1, usually multiple queries may be executed using the same plan. Therefore, RTQS may reuse a previously constructed plan for the new query. When no plan may be reused, the planner constructs a new one. (2) RTQS determines if a query meets its deadline using our schedulability analysis. The schedulability analysis is performed on the base station. If the query is schedulable, the parameters of the query are disseminated; otherwise, the query is rejected. (3)

At run-time the scheduler running on each node executes all admitted queries in a localized fashion.

In contrast to DCQS which does not support real-time communication, the key contribution of RTQS is the design and analysis of three *real-time* scheduling algorithms. Each scheduling algorithm achieves a different tradeoff between query prioritization and throughput. The *Nonpreemptive Query Scheduling* (NQS) algorithm achieves high throughput at the cost of priority inversion, while the *Preemptive Query Scheduling* (PQS) algorithm eliminates priority inversion at the cost of lower throughput. The *Slack-stealing Query Scheduling* (SQS) algorithm combines the benefits of NQS and PQS by improving the throughput while meeting all deadlines.

5.2.1 Constructing plans

A plan has two properties: (1) When the query involves aggregation, the plan must respect the precedence constraints introduced by aggregation: a node is a assigned to transmit in a later step then any of its children. We opted to respect these precedence constraints even for queries that do not involve aggregation because such an approach results in transmission schedules that have *long contiguous* periods of activity/inactivity: the node transitions from a sleep state to the active state just-in-time to receive the data from its children and transitions back to sleep after it completes collecting data from its children and relaying it to its parent. Schedules with long contiguous periods of activity/inactivity are efficient because they reduce the wasted energy in transitions between sleep and active states. (2) Each node is assigned in sufficient steps to transmit its entire data report. We use $T_l[i]$ to denote the set of transmissions assigned to step i ($0 \leq i < L_l$) in the plan of query l, where L_l is the length of the plan. Since we opted to have a node wait to to receive the data reports from its children (to support data aggregation and improve energy efficiency), the query latency may be reduced by assigning the transmissions of a node with a larger depth in the routing tree to an earlier step of the plan. This strategy reduces the query latency because it reduces the time a node waits for the data reports from all its children.

Figure 4.3 shows an IC graph and the plan constructed by the planner. The solid lines indicate the communication edges in the routing tree while the dashed lines indicate interference edges. Node a is the base-station. The plan in Figure 4.4 is constructed assuming that the data report generated by a node can be transmitted in a single step for each instance. The planner assigns conflict-free transmissions in each step. For example, transmissions $\overrightarrow{ne}$ and $\overrightarrow{po}$ are assigned to step $T_l[1]$ since they do not conflict with each other. The precedence constraints introduced by aggregation are respected. For example, nodes p and q are assigned in earlier steps than their parent o. In previous chapter we proposed a distributed algorithm for constructing plans based on the IC graph. Upon the completion of the algorithm each node knows in what steps it transmits and receives.

The plan of a query l depends on the IC graph, the set of source nodes, and the aggregation function. Query instances executed at different times may need different plans if the IC graph changes. However, to handle dynamics in channel conditions, RTQS can construct plans that are robust against certain variations in the IC graph (see Section 4.3). This allows instances executed at different times to be executed according to the same plan. Moreover, note that queries with the same aggregation function and sources but with different periods, start times, or priority can be executed according to the same plan. Furthermore, even queries with different aggregation functions may be executed according to the same plan. Let $W_l[i]$ be the number of slots node i needs to transmit its data report to its parent for an instance of query l. If the planner constructs a plan for a query l, the same plan can be reused to execute a query h if $W_l[i] = W_h[i]$ for all nodes i. Examples of queries that share the same plan are the queries for the maximum temperature and the average humidity in a building. For both queries a node transmits one data report in a single step (i.e., $W_{max}[i] = W_{avg}[i] = 1$ for all nodes i) if the slot size is sufficiently large to transmit a packet with two values. For the max query, the outgoing packet includes the maximum value of the data reports from itself and its children. For the average query, the packet includes the sum of the values and the number of data sources that contributed to the sum. We say that two queries belong to the same *query class* if they may be executed according to the same plan. Since queries with different temporal properties and aggregation functions may share a same plan, a WSN may only need to support a small number of query classes. This allows RTQS to amortize

the cost of constructing a query plan over many queries and effectively reduces the overhead.

5.2.2 Overview of Scheduling Algorithms

The scheduler executes a query instance according to the plan of its query. The scheduler improves the query throughput by overlapping the transmissions of multiple instances (belonging to one or more queries) such that: (1) All steps executed in a slot are conflict-free. Two steps of instances $I_{l,u}$ and $I_{h,v}$ are *conflict free* ($I_{l,u}.i \parallel I_{h,v}.j$) if all pairs of transmissions in $T_l[I_{l,u}.i] \cup T_h[I_{h,v}.j]$ are conflict free. (2) The steps of each plan are executed in order: if step $I_{l,u}.i$ is executed in slot s_i, step $I_{l,u}.j$ is executed in slot $s_j < s_i$ then $I_{l,u}.j < I_{l,u}.i$. This ensures that the precedence constraints required by aggregation are preserved.

The scheduler maintains a record of the start time, period, and priority of all admitted queries. Additionally, the scheduler knows the step numbers in which the host node is assigned to transmit or receive in each plan and the plan's length. RTQS supports both preemptive and nonpreemptive query scheduling.

A nonpreemptive scheduler controls only the *start* of an instance; once an instance starts executing, a nonpreemptive scheduler cannot preempt it. In contrast, a preemptive scheduler may *preempt* an instance to allow a higher priority instance to execute when the two cannot be executed concurrently.

We first consider a brute-force approach for constructing a preemptive scheduler: in every slot s, a brute-force scheduler would consider the released instances in order of their priority and execute all steps that do not conflict in s. Unfortunately, the time complexity of this approach is high, since each pair of steps must be checked for conflicts. Since the scheduler dynamically determines the steps executed in a slot, it must have low time complexity.

To reduce the time complexity of the scheduler we introduced the concept of *minimum step distance* in previous chapter. Let $I_{l,u}.i$ and $I_{h,v}.j$ be two steps in the plans of any instances $I_{l,u}$ and $I_{h,v}$, respectively. We define the *step distance* between $I_{l,u}.i$ and $I_{h,v}.j$ as $|I_{l,u}.i - I_{h,v}.j|$. The *minimum step distance* $\Delta(l, h)$ is the smallest step

distance between $I_{l,u}$ and $I_{h,v}$ such that the two steps $I_{l,u}.i$ and $I_{h,v}.j$ may be executed concurrently without conflict:

$$|I_{l,u}.i - I_{h,v}.j| \geq \Delta(l,h) \Rightarrow I_{l,u}.i \parallel I_{h,v}.j$$
$$\forall I_{l,u}.i < L_l, I_{h,v}.j < L_h$$

where, L_l and L_h are the lengths of the plans of queries l and h, respectively. Therefore, to ensure that no conflicting transmissions are executed in a slot, it is sufficient to enforce a minimum step distance between any two steps.

The minimum step distance captures the degree of parallelism that may be achieved due to spatial reuse in a multi-hop WSN. For simplicity consider the case when $L = L_q = L_h$. In the worst case, when $\Delta(l,h) = L$, a single instance is executed in the network at a time. If $\Delta(l,h) = L/2$, then two instances can be executed in the network at the same time. A distributed algorithm for computing $\Delta(l,h)$ is presented in [28]. The minimum step distance $\Delta(l,h)$ depends on the IC graph and the plans of l and h. The number of minimum step distances that a scheduler stores is quadratic in the number of plans. Two pairs of queries (l,h) and (m,n) have the same minimum step distance if (l,m) and (h,n) have the same plan. Therefore, in practice the number of minimum step distances that must be stored the memory cost is small since the planner uses only few plans. For clarity, we first present the scheduling algorithms assuming that all queries are executed according to a single plan of length L in this section. In this case, the scheduler maintains a single minimum step distance Δ. We extend our algorithms to handle queries with different plans in the next section.

5.2.3 Nonpreemptive Query Scheduling

To efficiently enforce the minimum step distance for NQS, we take advantage of the fact that once an instance is started, it cannot be preempted. As such, the earliest time at which an instance $I_{l,u}$ may start (i.e., execute step $I_{l,u}.i = 0$) is after the previous instance $I_{h,v}$ completes step $I_{h,v}.j = \Delta - 1$ (since $|\Delta - 0| \geq \Delta$). Since the execution of $I_{l,u}$ and $I_{h,v}$ cannot be preempted, if we enforce the minimum step

distance between the start of the two instances then their concurrent execution is conflict-free for their remaining steps since steps $I_{l,u}.i = x$ and $I_{h,v}.j = x + \Delta$ are executed in the same slot and $|(x + \Delta) - x| \geq \Delta$. Therefore, to guarantee that a nonpreemptive scheduler executes conflict-free transmissions in each slot, it suffices to enforce a minimum step distance of Δ between the start times of any two instances.

NQS maintains two queues: a *run* queue and a *release* queue. The *release* queue is a priority queue containing all instances that have been released but are not being executed. The *run* queue is a FIFO queue and contains the instances to be executed in slot s. Although the *run* queue may contain multiple instances, a node is involved in transmitting/receiving for at most one instance (otherwise, it would be involved in two conflicting transmissions). A node n determines if it transmits/receives in slot s by checking if it is assigned to transmit/receive in any of the steps to be executed in slot s. If a node does not transmit or receive in slot s, it turns off its radio for the duration of the slot.

NQS enforces a minimum step distance of at least Δ between the start times of any two instances by starting an instance in two cases: (1) when there is no instance being executed (i.e., *run*=$\emptyset$) and (2) when the step distance between the head of the *release* queue (i.e., the highest priority instance that has been released) and the tail of the *run* queue (i.e., the last instance that started) is larger Δ. When an instance starts, it is moved from the *release* queue to the *run* queue.

Consider the example shown in Figure 5.3(a) where three queries, Q_{hi}, Q_{med} and Q_{lo} are executed according to the shown workload parameters. Each query is executed according to the same plan of length $L = 15$ and minimum step distance $\Delta = 8$. We assign higher priority to queries with tighter deadlines. The upward arrows indicate the release time of an instance. I_{lo} (in the example we drop the instance count since it is always zero) is released and starts in slot 0 since no other instance is executing (*run*=$\emptyset$). The first instances of Q_{med} and Q_{hi} are released in slots 2 and 6, respectively. However, neither may start until slot 8 when I_{lo} completes 8 steps (i.e., when $I_{lo}.i = 8 \geq \Delta$) resulting in priority inversions. I_{hi} then starts at slot 8 since it is the highest priority instance in *release*. Similarly, in slot 16, NQS starts I_{med} after I_{hi} completes $\Delta = 8$ steps. NQS continues to construct the schedule in figure.

```
event: new instance I_{l,u} is released
    if (run=∅) then start(I_{l,u})
    else release = release ∪ {I_{l,u}}
event: start of new slot s
    if (release ≠ ∅)
      let I_{l,u} be the highest priority instance in release
      if (Last_{q',k'}.i ≥ Δ) then
        start(I_{l,u})
        Last_{q',k'} = I_{l,u}
    for each I_{l,u} ∈ run execute-step(I_{l,u})

start(I_{l,u}):
    run = run ∪ {I_{l,u}}
execute-step(I_{l,u}):
    determine if node should send/recv in I_{l,u}.i
    I_{l,u}.i = I_{l,u}.i + 1
    if I_{l,u}.i = L_q then run = run \ {I_{l,u}}
```

Figure 5.1: NQS pseudocode

When a new instance is released, NQS inserts it in the *release* queue. Since the *release* queue is a priority queue keyed by the priority of a query instance, this operation takes $O(\log |release|)$. In each slot, NQS determines what instances should start executing. This operation takes constant time, since it involves comparing the step distance between the instances at the head of *release* queue and tail of *run* queue with the minimum step distance. To determine if a node should send, receive, or sleep, NQS iterates through the instances in the *run* queue. This requires $O(|run|)$ time if each node maintains a bit vector indicating whether it transmits, receives, or sleeps in each step of a plan. Thus, the complexity of the operations performed in a slot is $O(|run|)$.

5.2.4 Preemptive Query Scheduling

A drawback of NQS is that it introduces priority inversions. To eliminate prioritization inversion, we devised PQS which preempts the instances that conflict with the execution of a higher priority instance. A key feature of PQS is a new and efficient mechanism for enforcing the minimum step distance that supports preemption. To

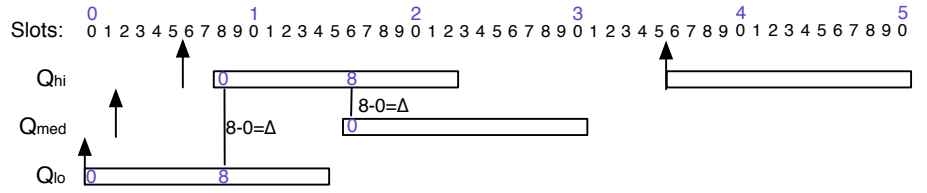
```
event: new instance I_{l,u} is released
       release = release ∪ {I_{l,u}}
event: start of new slot s
     for each I_{l,u} ∈ release
       if (may-resume(I_{l,u}) = true) then resume(I_{l,u})
     for each I_{l,u} ∈ run
       execute-step(I_{l,u})

resume(I_{l,u}):
     run = run ∪ {I_{l,u}}; release = release − {I_{l,u}}
     add I_{l,u} to all mayConflict[x] such that |I_{l,u}.i − x| < Δ
preempt(S):
     run = run − S; release = release ∪ S
     remove I_{l,u} from all mayConflict
may-resume(I_{l,u}):
     if (mayConflict[I_{l,u}.i] = ∅) then return true
     if (I_{l,u} has higher priority all instances in mayConflict[I_{l,u}.i])
       preempt(mayConflict[I_{l,u}.i]); return true
     return false
execute-step(I_{l,u}):
     determine if node should send/recv in I_{l,u}.i
     I_{l,u}.i = I_{l,u}.i + 1
     if I_{l,u}.i = L then run = run − {I_{l,u}}
     mayConflict[I_{l,u}.i − Δ]=mayConflict[I_{l,u}.i − Δ] − {I_{l,u}}
     mayConflict[I_{l,u}.i + Δ]=mayConflict[I_{l,u}.i + Δ] ∪ {I_{l,u}}
```

Figure 5.2: PQS pseudocode

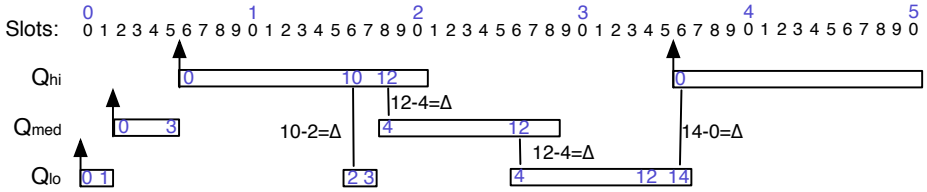
enforce the minimum step distance PQS maintains L_q *mayConflict* sets. Each *mayConflict*$[x]$ set contains the instances which are in the *run* queue and conflict with *any* instance executing step x in its plan:

$$mayConflict[x] = \{I_{h,v} \in run || x - I_{h,v}.i| < \Delta\}$$

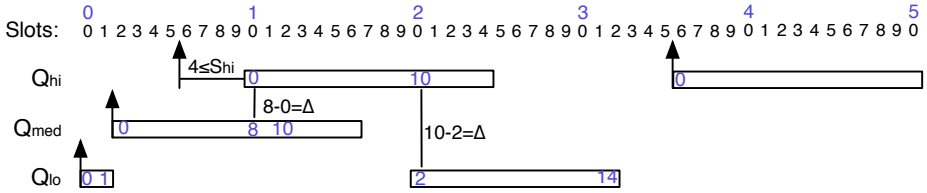
PQS (see Figure 5.2) maintains a *run* queue and a *release* queue which are keyed by the query instance priority. When a new instance is released, it is added to the *release* queue.



(a) Schedule constructed by NQS



(b) Schedule constructed by PQS



(c) Schedule constructed by SQS

Figure 5.3: Scheduling with different prioritization policies. Workload: P_{hi}=30, D_{hi} =20,P_{med}=65, D_{med} =28, P_{lo}=93, D_{lo} =93.

PQS starts/resumes an instance $I_{l,u}$ ($I_{l,u} \in$ *release*) in two cases. (1) If the next step $I_{l,u}.i$ of $I_{l,u}$ may be executed concurrently with all instances in the *run* queue without conflict, PQS starts/resumes it. To determine if this is the case, it suffices for PQS to check if $mayConflict[I_{l,u}.i]$ is empty. When an instance is started or resumed, it is moved from the *release* queue to the *run* queue. The membership of $I_{l,u}$ in the *mayConflict* sets is updated to reflect that $I_{l,u}$ is executed in the current slot: $I_{l,u}$ is added to all $mayConflict[x]$ sets such that $|I_{l,u}.i - x| < \Delta$ since the execution of any of those steps would conflict with the execution of step $I_{l,u}.i$. (2) $I_{l,u}$ is also started/resumed if it has higher priority than all the instances in $mayConflict[I_{l,u}.i]$ since otherwise there will be a priority inversion. For $I_{l,u}$ to be executed without conflict, all instances in $mayConflict[I_{l,u}.i]$ must be preempted. When an instance is preempted, it is moved from the *run* queue to the *release* queue and it is removed from all *mayConflict* sets. As in case (1), $I_{l,u}$ is added to all $mayConflict[x]$ sets such that $|I_{l,u}.i - x| < \Delta$.

After an instance executes a step, its membership in the *mayConflict* sets must also be updated. Since step $I_{l,u}.i$ is executed in slot s, in the next slot (when $I_{l,u}$ executes step $I_{l,u}.i+1$) $I_{l,u}$ will not conflict with an instance executing step $I_{l,u}.i-\Delta$ but will conflict with an instance executing step $I_{l,u}.i+\Delta$. Accordingly, $I_{l,u}$ is removed from $mayConflict[I_{l,u}.i-\Delta]$ and added to $mayConflict[I_{l,u}.i+\Delta]$.

Figure 5.3(b) shows the schedule of PQS for the same workload used in the example for NQS. Instance I_{lo} starts in slot 0 since no other instances have been released ($mayConflict[0]=\emptyset$). I_{med} is released in slot 2. Since $mayConflict[0]=\{I_{lo}\}$ and I_{med} has higher priority than I_{lo}, PQS preempts I_{lo}. Consequently, I_{lo} is removed from the *run* queue and all *mayConflict* sets, and it is added to the *release* queue. I_{med} is added to *run* queue and to all $mayConflict[x]$ sets where $0 \leq x < 8$. I_{hi} is released in slot 6. Since $mayConflict[0] = \{I_{med}\}$ and I_{hi} has higher priority then I_{med}, PQS preempts I_{med} and starts I_{hi}. The *mayConflict* sets are updated accordingly. An interesting case occurs in slot 16, when I_{hi} executes step 10. At this point, $mayConflict[2] = \emptyset$ since I_{med} was preempted and I_{hi} completed 10 steps ($|10-2| \geq 8$). As a result, I_{lo} may execute step 2 in its plan while I_{hi} executes step 10 without conflict. I_{hi} and I_{lo} are executed concurrently until step 18 because their step distance exceeds the minimum step distance. In the beginning of slot 18, $mayConflict[4]=\{I_{lo}\}$. Note that I_{hi} is not a member of this set since $|12-4| \geq 8$. Since the step counter of I_{med} is 4 and I_{med} has higher priority than I_{lo}, PQS preempts I_{lo} and resumes I_{med}. PQS then updates the conflict sets by removing I_{lo} from all of them and adding I_{med} to $mayConflict[x]$ sets where $|x-4| < 8$. I_{lo} resumes in slot 26 when $mayConflict[4]$ becomes empty. The example shows that by eliminating priority inversion PQS achieves lower latencies for I_{hi} and I_{med} than NQS. However, the query throughput is lower because it allows less overlap in the execution of instances. This exemplifies the tradeoff between prioritization and throughput in query scheduling. In the next section, we will characterize this tradeoff analytically.

When an instance is released, it is added to the *release* queue which takes $O(\log |release|)$ time. In every slot, PQS iterates through the instances in *release* to determine if they may be resumed. If we organize the *mayConflict* sets as balanced trees keyed by instance priority, the time complexity of this operation is $O(|release| \cdot \log |run|)$. We note that the **resume** and **preempt** functions take constant time since an instance $I_{l,u}$ may be a member of at most 2Δ *mayConflict* sets and Δ does not depend on

the number of instances in *release* or *run*. Similar to NQS, $O(|run|)$ is necessary for a node to determine if it transmits, receives, or sleeps in a slot. Thus, the time complexity of operations performed per slot is $O(|release| \cdot \log |run| + |run|)$.

5.2.5 Analysis of NQS and PQS

In this section we present worst-case response analyses for PQS and NQS. The worst-case response time of a query is the maximum query latency of any of its instances. Our analysis can be used for admission and rate control at the base station when a query is submitted. In our analysis we assume that the deadlines are shorter than the periods.

Analysis of NQS. Since NQS is non-preemptive, the response time R_l of query l is the sum of its plan's length L and the worst-case delay W_l that any instance experiences before it is started: $R_l = W_l + L$. Note that for convenience we use the slot size as the time unit.

To compute W_l, we construct a recurrent equation similar to the response time analysis for processor scheduling [8]. Consider an instance I_l. Note that for clarity we drop the instance index from the instance notation in our analysis. Since NQS is a non-preemptive scheduling algorithm, to compute the response time of a query l we must compute the worst-case interference of higher priority instances and the maximum blocking time of l due to the nonpreemptive execution of lower priority instances. Our analysis is based on the following two properties.

Theorem 4 *An instance is blocked for at most $\Delta - 1$ slots.*

Proof 4 *Consider the following two cases based on when an instance I_l is released. (1) If all executing lower priority instances have completed at least Δ steps, NQS starts I_l without blocking. (2) If a lower priority instance which did not complete Δ steps is executing, I_l is blocked. Note that there can be only one lower priority instance that blocks I_l, because the interval between the starting times of two consecutive instances must be at least Δ. Hence there can only be one executing instance that has not completed Δ steps when I_l is released. The longest blocking time occurs when the low*

priority instance has completed one step when I_l is released. In this case I_l is blocked for $\Delta - 1$ slots.

Theorem 5 *A higher priority instance interferes with a lower priority instance for at most Δ slots.*

Proof 5 *NQS starts the highest priority instance when the last started instance has completed at least Δ steps. Therefore, every high priority instance delays the execution of a low priority instance by at most Δ slots. The worst-case interference occurs when the lower and higher priority instances are released simultaneously.*

The number of instances of a higher priority query h that interfere with I_l is upper-bounded by $\lceil \frac{W_l}{P_h} \rceil$. Therefore, the worst-case delay that I_l experiences before it starts is:

$$W_l = (\Delta - 1) + \sum_{h \in hp(l)} \left\lceil \frac{W_l}{P_h} \right\rceil \cdot \Delta \tag{5.1}$$

where $hp(l)$ is the set of queries with priority higher than or equal to l's priority. W_l can be computed by solving (5.1) using a fixed point algorithm similar to that of the response time analysis [8].

Note that our analysis differs from the classical processor response time analysis in that multiple transmissions may occur concurrently without conflict in a WSN due to spatial reuse of the wireless channel. This is captured in our analysis in that a higher priority instance may delay a lower priority instance by at most Δ, which is usually smaller than the execution time of the instance (i.e., the plan's length L).

Analysis of PQS. A higher priority instance cannot be blocked by a lower priority instance under PQS[9]. We observe that after an instance completes Δ steps, no newly released instance will interfere with its execution because their step distance would

[9]Our analysis assumes that every instance is released in the beginning of a slot, which is the time granularity of our scheduling algorithms. Strictly speaking, a higher priority instance may still be blocked by at most one slot. This blocking term can be easily incorporated into our analysis.

be at least Δ, allowing them to execute concurrently. Therefore, we split I_l into two parts: a preemptable part of length Δ and nonpreemptable part of length $L - \Delta$. Higher priority instances may interfere with I_l only during its preemptable part. Thus, the response time of a query l is the sum of response time of the preemptable part R'_l and the length of the nonpreemptable part: $R_l = L - \Delta + R'_l$.

A query h with higher priority than l interferes with l for at most $\lceil \frac{R'_l}{P_h} \rceil \cdot C_{max}(l,h)$ slots, where $C_{max}(l,h)$ is the worst-case interference of an instance of h on an instance of l. Thus, worst-case response time of the preemptable part of l is:

$$R'_l = \Delta + \sum_{h \in hp(l)} \left\lceil \frac{R'_l}{P_h} \right\rceil \cdot C_{max}(l,h) \tag{5.2}$$

After finding the worst-case interference, R'_l may be computed by solving (5.2) using a fixed point algorithm similar to the one used in the response time analysis [8]. Next, we determine the worst-case interference.

Theorem 6 *An instance I_l is interfered by a higher priority instance I_h for at most $C_{max}(l,h) = \min(2\Delta, L)$ slots.*

Proof 6 *We analyze I_h's interference on I_l in the following cases. (1) If I_h is released no later than I_l, then I_h's interference on I_l is at most Δ, since I_l may start when I_h completes Δ steps.*
(2) If I_h is released while I_l is executing its nonpreemptable part, the interference is zero.
(3) If I_h is released while I_l is executing its preemptable part, I_h preempts I_l. Let x be the number of steps I_l has completed, when I_h preempts it. We note that $x \leq \Delta$ since I_l is executing its preemptable part. There are three sub-cases. (3a) If I_h is not preempted by any higher priority instance, then I_l will be resumed after I_h completes $\Delta + x$ steps to enforce the minimum step distance between I_l and I_h. Thus, the interference is $C = \Delta + x$. If I_h is preempted after executing $y \leq \Delta$ steps we must consider two cases as illustrated in Figure 5.4. Recall that plans start with step 0. (3b) If $x \geq y$, PQS resumes I_h before I_l due to the minimum step distance constraint. In this case, I_h's interference on I_l is $C = \Delta + x$. (3c) If $x < y$, then I_l is resumed before

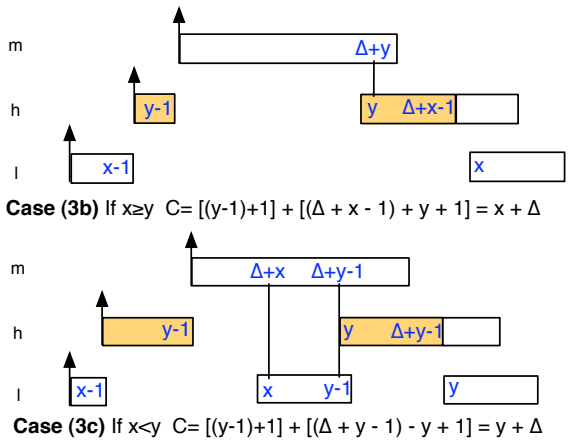


Figure 5.4: Interference of I_h on I_l under PQS.

I_h and it may execute up to $(x - y)$ steps until I_h is resumed. Thus, I_h's interference on I_l is $C = \Delta + y$.

From all the above cases, I_h's worst-case interference on I_l is $C = \Delta + \max(x, y)$. Since $x \leq \Delta$ and $y \leq \Delta$, then $C_{max} \leq 2\Delta$. However, when $L < 2\Delta$, I_h finishes before I_l reaches 2Δ; in this case the interference is only L. Thus, I_h's worst-case interference on I_l is $C_{max} = \min(2\Delta, L)$.

It is important to note that preempting an instance results in higher interference than the nonpreemptive case. As shown in the above proof, the interference in the preemptive case is $C = \Delta + \max(x, y)$ compared to Δ in the nonpreemptive case. Therefore, preemption incurs $\max(x, y)$ slots of additional interference compared to the no preemption case. The additional interference in the preemptive case results in a lower degree of concurrency and hence lower query throughput. This shows the inherent trade-off between prioritization and throughput in conflict-free query scheduling.

5.2.6 Slack Stealing Query Scheduling

SQS combines the benefits of NQS and PQS in that it improves query throughput while meeting all deadlines. The design of SQS is based on the observation that preemption lowers throughput, and hence it should be used only when necessary

for meeting deadlines. We define the *slack* of a query l, S_l, to be the maximum number of slots that an instance of l allows a lower priority instance to execute before preempting it. SQS has two components: an admission algorithm and a scheduling algorithm. The admission algorithm runs on the base station and determines the slack and schedulability of each query when it is issued. The scheduling algorithm executes admitted queries based on their slacks.

SQS Scheduler. SQS may start an instance $I_{h,v}$ in any slot in the interval $[r_{h,v}, r_{h,v}+S_h]$, where S_h is the slack of query h and $r_{h,v}$ is the release time of the v^{th} instance of h. Intuitively, SQS can dynamically determine the best time within the interval to start $I_{h,v}$ such that $I_{h,v}$'s interference on lower priority instances is reduced. Since a lower priority instance $I_{l,u}$ is not interfered by $I_{h,v}$ if $I_{l,u}$ has completed at least Δ steps, SQS postpones the start of the higher priority instance $I_{h,v}$ if the lower priority instance $I_{l,u}$ has completed at least $\Delta - S_h$ steps. An advantage of the slack stealing approach is that it opportunistically avoids preemption and the related throughput reduction when allowed by query deadlines.

SQS requires a minor modification to PQS. Specifically, we change how the release of an instance $I_{h,v}$ is handled. If *mayConflict*[0] is empty, $I_{h,v}$ is released immediately. If SQS determines that all the instances in *mayConflict*[0] have completed at least $\Delta - S_h$ steps, SQS delays $I_{h,v}$ until the lower priority instances complete Δ steps in their plans (i.e., when *mayConflict*[0] becomes empty). All instances whose release is delayed are maintained in a *pending* queue. If $I_{h,v}$ does not have sufficient slack to allow the lower priority instances to complete Δ steps, then SQS (1) preempts all instances in *mayConflict*[0], (2) resumes the highest priority instance in the *release* or *pending* queues (which is not necessarily $I_{h,v}$), and (3) moves all instances from the *pending* queue to the *release* queue.

Figure 5.3(c) shows the schedule under SQS with the example workload. Assume that the admission algorithm of SQS determined that Q_{hi} and Q_{med} have slacks $S_{hi} = 5$ and $S_{med} = 2$, respectively. I_{lo} is released and starts its execution in slot 0. I_{med} is released in slot 2. SQS preempts I_{lo}, because even if I_{med} would be postponed for $S_{med} = 2$ slots, I_{lo} would not complete $\Delta = 8$ steps. I_{hi} is released in slot 6. SQS decides to continue executing I_{med} because in $4 \leq S_{hi}$ slots, I_{med} will complete executing $\Delta = 8$ steps, i.e., SQS avoids preempting I_{med} by allowing it to steal 4 slots

from I_{hi}. SQS uses preemption in slot 2 but not in slot 6. This highlights that SQS can adapt preemption decisions to improve throughput while meeting all deadlines.

Admission Algorithm. The admission algorithm determines the schedulability and slacks of queries. It considers queries in decreasing order of their priorities. For each query, it performs a binary search in $[0, \Delta]$ to find the maximum slack that allows the query to meet its deadline. Note that there is no benefit for a lower priority instance to steal more than Δ slots from a higher priority instance since they may be executed in parallel when their step distance is at least Δ. The admission algorithm tests whether the query can meet its deadline by computing its worst-case response time as a function of the slack. If the query is unschedulable with zero slack, it is rejected; otherwise, it is admitted.

To compute the worst-case response time of a query we split a query instance into two parts: a preemptable part and a nonpreemptable part. Under PQS, the preemptable part is Δ slots. In contrast, under SQS, an instance I_l may steal from a higher priority instance at least $m_l = \min_{x \in hp(l)} S_x$ steps. Thus, the length of the preemptable part is at most $\Delta - m_l$ slots under SQS; the length of the nonpreemptable part is therefore $L - (\Delta - m_l)$ slots. Hence, the worst-case response time of query l with slack S_l is:

$$R_l(S_l) = L - (\Delta - m_l) + R'_l(S_l) \tag{5.3}$$

where R' is the worst-case response of time the preemptable part.

Theorem 7 *Under SQS, an instance I_l may be interfered by a higher priority instance I_h for at most $C_{max} = \min(2\Delta - m_l, L)$ slots, where $m_l = \min_{x \in hp(l)} S_x$.*

Proof 7 *We initially assume $L > 2\Delta - m_l$. Similar to PQS the worst-case interference occurs when a higher priority instance is released during I_l's preemptable part. In this case, I_l either (1) steals slack from one or more higher priority instances or (2) does not steal slack from any higher priority instance.*
(1) When I_l steals slack we consider the following two sub-cases depending on whether I_l successfully steals enough slack to complete Δ steps.
(1a) I_l completes Δ steps without being preempted. In this case I_h's interference on

I_l is zero.
(1b) Otherwise, I_l is preempted after executing x steps by a higher priority instance I_m (not necessarily I_h). Next, we show that the execution of I_m does not affect I_h's interference on I_l. As a result, it would be sufficient to only consider the case when I_h itself preempts I_l. We note that I_m must have a higher priority than I_h since SQS always resumes the highest priority instance in release *when an instance is preempted. I_h's interference on I_l is not affected by I_m if neither I_l nor I_h execute while I_m executes its preemptable part (i.e., the relative phasing of I_l and I_h remains the same). I_h cannot execute because it cannot start before I_m completes Δ steps (due to minimum step distance). Note that I_l cannot steal slack from I_m as I_l is in* release*. I_l cannot execute as I_h must be started before I_l resumes (since I_h's next step is 0, I_l's next step is $x > 0$, and hence the step distance between I_m and I_h is higher than that between I_m and I_l). Since, I_h cannot start before I_m completes Δ steps, I_l also cannot start before I_m completes Δ steps.*
We now consider the case when I_h is the instance that preempts I_l. Similar to Theorem 6 we consider sub-cases depending on whether I_h is preempted. If I_h is not preempted, according to the proof of Theorem 6, I_h's interference on I_l is $C = \Delta + x$. However, unlike in PQS where $x < \Delta$, for SQS we have a tighter bound on x: $x < \Delta - m_l$. Hence, I_h's interference on I_l is $C_{max} = 2\Delta - m_l$. If I_h is preempted by a higher priority instance, let y be the number of steps I_h has completed before it is preempted. We note that $y < m_l$, since m_l is the smallest slack of any query whose priority is higher or equal to l. Similar to PQS, the worst-case interference in the two cases is: $C(x) = \Delta + \max(x, y)$. However, unlike PQS, we have tighter bounds on x and y: $x < m_l$ and $y < m_l$. Thus, the worst-case interference of I_h on I_l is $C_{max} = 2\Delta - m_l$.
(2) In this case I_l is preempted by I_h. This case is handled similarly to (1b).
Similar to PQS, when $L < 2\Delta - m_l$ the interference cost is reduced L. Therefore the worst-case interference of I_h on I_l is $\min(2\Delta - m_l, L)$.

To compute R'_l we must account for the jitter introduced by slack stealing, i.e., a higher priority instance I_h may delay its start by at most S_h. Accordingly, R' is:

$$R'_l(S_l) = (\Delta - m_l) + S_l + \sum_{h \in hp(l)} \left\lceil \frac{R'_l(S_l) + S_h}{P_h} \right\rceil \cdot C_{max}(l, h)$$

where, $\Delta - m_l$ is the maximum length (execution time) of the preemptable part, S_l is the maximum time interval when I_l may be blocked by a lower priority instance due to slack stealing, and $C_{max}(l,h) = \min(2\Delta - m_l, L)$ is the worst-case interference when slack stealing is used.

5.3 Handling Multiple Plans

So far our algorithms and their schedulability analysis were presented under the assumption that all queries belong to the same class i.e., they are executed according to the same plan. In this section, we consider the case when there are multiple query classes. We define the *minimum step distance between two queries classes c and c′* $\Delta(c, c')$ as the minimum number of slots a query instance of class c' must wait after a query instance of class c started such that there are no conflicts. Note that Δ is *not* commutative.

5.3.1 Multi-class NQS

When all queries belong to a single query class, NQS only needs to check if the step distance between the highest priority instance in the *release* queue and the instance at the tail of the *run* queue exceeds the minimum step distance to guarantee conflict-free transmissions. However, in the case of multiple query classes, to guarantee that *all* minimum step distances are enforced, NQS should check if the step distance between highest priority instance in the *release* queue and *all* instances in *run* queue exceeds the minimum step distances between their respective query classes. An efficient mechanism for doing this is for the NQS scheduler to keep track of the slot when the last instance of each query class started. To enforce *all* minimum step distances it suffices to record the time when the last instance of each class started. Using this information, NQS ensures that any new instance that is released is started only when the its step distances to all other instances currently being executed exceeds their respective minimum step distances.

We note that NQS is not work-conserving: NQS does not start a released lower priority instance I_l if there is a higher priority instance I_h, even when the start of I_l would not conflict with any of the instances currently being executed. We made this design choice to bound the time when priority inversions may occur: at most a single lower priority instance blocks the execution of a higher priority instance.

To handle multiple classes, we must store additional information: (1) for each pair for query classes we must maintain their minimum step distances and (2) an additional integer per query class to keep track of when the last instance of query class started. The number of comparisons necessary to enforce the minimum step distances equals the number of query classes. As a consequence, the NQS scheduler handles multiple classes without increasing its computational complexity since the number of classes is a constant (i.e., it does not depend on the number of instances either in *release* or in *run* queues).

Schedulability Analysis. For clarity of presentation, in the schedulability analysis of NQS and subsequent multi-class schedulers we use interchangeably h and $cls(h)$ in $\Delta(cls(l), cls(h))$ to denote the minimum step distance between the query classes of queries l and h.

To extend the schedulability analysis of NQS to the multi-class case, we must determine the impact of having multiple classes on the the blocking and interference terms (see Property 4 and Property 5, respectively). The maximum blocking time an instance of a query l suffers due to a lower priority instance of a query m is:

$$B_l = \max_{m \in lp(l)} \Delta(m, l) - 1 \tag{5.4}$$

Proving that the Equation 5.4 holds follows a similar argument as the proof of Property 4. The worst-case blocking of an instance of query l occurs when a lower priority instance of a query m for which $\Delta(m, l) = B_m + 1$ starts one step before l's instance is released.

The multi-class NQS scheduler starts an instance I_l after all instances I_h previously started complete $\Delta(h, l)$ steps. Thus, the worst-case interference of a higher priority instance I_h on I_l is at most $I_l(h) = \max_{m \in hp(l)} \Delta(h, m)$. Thus, the worst-case response

```
event: new instance I_{l,u} is released
       release = release ∪ {I_{l,u}}
event: start of new slot s
    for each I_{l,u} ∈ release
      if (may-resume(I_{l,u}) = true) then resume(I_{l,u})
    for each I_{l,u} ∈ run
      execute-step(I_{l,u})

resume(I_{l,u}):
    run = run ∪ {I_{l,u}}; release = release − {I_{l,u}}
    for each c ∈ C
       add I_{l,u} to all mayConflict[x][c] such that
          |I_{l,u}.i − x| < Δ(l,c)
preempt(S):
    run = run − S; release = release ∪ S
    remove I_{l,u} from all mayConflict sets
may-resume(I_{l,u}):
    if (mayConflict[I_{l,u}.i][cls(l)] = ∅) then return true
    if (I_{l,u} has higher priority instances in mayConflict[I_{l,u}.i][cls(l)])
      preempt(mayConflict[I_{l,u}.i]); return true
    return false
execute-step(I_{l,u}):
    determine if node should send/recv in I_{l,u}.i
    I_{l,u}.i = I_{l,u}.i + 1
    if I_{l,u}.i = L(l) then run = run − {I_{l,u}}
    for each c ∈ C
      mayConflict[I_{l,u}.i − Δ(l,c)][c] = mayConflict[I_{l,u}.i − Δ(l,c)][c] − {I_{l,u}}
      mayConflict[I_{l,u}.i + Δ(l,c)][c] = mayConflict[I_{l,u}.i + Δ(l,c)][c] ∪ {I_{l,u}}
```

Figure 5.5: Pseudocode of multi-class PQS schedule.

time of a query l is $R_l = L_l + W_l$ where L_l is the length of plan of query class l and W_l is the worst-case delay an instance of l observes before it starts:

$$W_l = B_l + \sum_{h \in hp(l)} \left\lceil \frac{W_l}{P_h} \right\rceil \cdot I_l(h) \tag{5.5}$$

5.3.2 Multi-class PQS

To extend PQS to multiple classes, we need to extend the definition of the *mayConflict* sets. We define $mayConflict[x][c]$ to be the set that contains the instances which are

in the *run* queue and conflict with *any* instance executing step x in the plan of class c:

$$\begin{aligned} &\mathit{mayConflict}[x][c]=\{I_{h,v} \in \mathit{run} \mid x - I_{h,v}.i < \Delta(h,c) \text{ and} \\ &\qquad h \text{ is started/resumed earlier than any instance of class } c\} \end{aligned}$$

The functions used by PQS (*resume*, *may-resume*, and *execute-step*) need to be updated (see Figure 5.5). In updating PQS to handle multiple classes, particular attention must be paid to the *order* of arguments used in the minimum step distances since Δ is not commutative.

According to the definition of the *mayConflict* sets, the multi-class PQS scheduler determines the instances which may interfere with an instance $I_{l,u}$ executing step $I_{l,u}.i$ by inspecting the set $\mathit{mayConflict}[I_{l,u}.i][cls(l)]$. Accordingly, PQS will start/resume $I_{l,u}$ if either (1) the $\mathit{mayConflict}[I_{l,u}.i][cls(l)]$ set is empty or (2) all its members have lower priority than $I_{l,u}$. These changes are reflected in the *may-resume* function.

When an instance is started/resumed, it must be added to the appropriate *mayConflict* set in the *resume* function. This entails adding $I_{l,u}$ to all sets $\mathit{mayConflict}[x][c]$ such that $|I_{l,u}.i - x| < \Delta(l,c)$ and c is a query class. When a step in a plan is executed, the membership in the *mayConflict* sets is updated in the *execute-step* function. For any given class c, when an instance $I_{l,u}$ completes executing a step, it is removed from $\mathit{mayConflict}[I_{l,u}.i - \Delta(l,c)][c]$ and added to $\mathit{mayConflict}[I_{l,u}.i + \Delta(l,c)][c]$.

The multi-class PQS scheduler maintains $\sum_{c \in C} L(c)$ *mayConflict* sets, where $L(c)$ is the length of the plan for class c. Since c is a constant, the time complexity of the multi-class PQS is same as in the single class case.

Schedulability Analysis. To extend the analysis to the multi-class case, we follow the same approach as in the single query class case. We start by observing that once a query executes more than $E_l = \max_{m \in hp(l)} \Delta(l,m)$, no other query instance may preempt its execution. We split the execution of a query in two parts, a preemptable part of length E_l and a non-preemptable part of length $L_l - E_l$. Thus, the response time of a query is the sum of the response time of the preemptable part R'_l and the length of the non-premptable part: $R_l = L_l - E_l + R'_l$

The response time of the preemptable part is:

$$R'_l = E_l + \sum_{h \in hp(l)} \left\lceil \frac{R'_l}{P_h} \right\rceil \cdot C_{max}(l,h) \tag{5.6}$$

Theorem 8 *An instance I_l is interfered by a higher priority instance I_h for at most $C_{max}(l,h) = \min(\Delta(h,l) + \Delta(l,h)), L_l)$ slots.*

Proof 8 *We analyze I_h's interference on I_l in the following cases:*
(1) If I_h is released while I_l is executing its nonpreemptable part, the interference is zero.
(2) I_h is released no later than I_l, then I_h's interference on I_l is at most $\Delta(h,l)$, since I_l may start when I_h completes $\Delta(h,l)$ steps. Thus, $C(l,h) = \Delta(h,l)$.
(3) If I_h is released while I_l is executing its preemptable part, I_h preempts I_l. Let x be the number of steps I_l has completed, when I_h preempts it. We note that $x \leq \Delta(l,h)$ since I_l otherwise both I_l and I_h may be executed concurrently without conflict. There are three sub-cases to be considered: (a) I_h is not preempted by a higher priority instance, (b) I_h is preempted by a higher priority instance and I_l is not resumed before I_h, and (c) I_h is preempted by a higher priority instance and I_l is resumed before I_h.
(3a) If I_h is not preempted by any higher priority instance, then I_l will be resumed after I_h completes $\Delta(h,l) + x$ steps to enforce the minimum step distance between I_l and I_h. Thus, the interference is $C(l,h) = \Delta(h,l) + x \leq \Delta(h,l) + \Delta(l,h)$.
(3b) If I_h is preempted by I_m after it completes y steps and I_l cannot be resumed before I_h is. We know that $y < \Delta(h,l)$ since otherwise I_l and I_h may be executed concurrently. The earliest time when I_l may be resumed is when I_m completes at least $\Delta(m,l) + x$ steps and I_h completes $\Delta(m,h) + y$ steps. Depending on the relationship between x and y there are two possible interference patterns as shown in Figure 5.6. If $x \geq y$ (see case 3b-i), then the interference is $C(l,h) = y \leq \Delta(h,l)$, since $y \leq \Delta(h,l)$. If $x < y$ (see case 3b-ii), then the interference $C(l,h) = \Delta(h,l) + x \leq \Delta(h,l) + \Delta(l,h)$ since $x \leq \Delta(l,h)$.
(3c) If I_h is preempted by I_m, PQS may resume I_l before I_h since it resumes a lower priority instance as soon as it does not conflict with any higher priority instance. As such, the earliest time I_l may be resumed, is after I_m completes $\Delta(m,l) + x$ steps. I_l will be executed until step x' when I_h may be resumed without conflicting with I_m

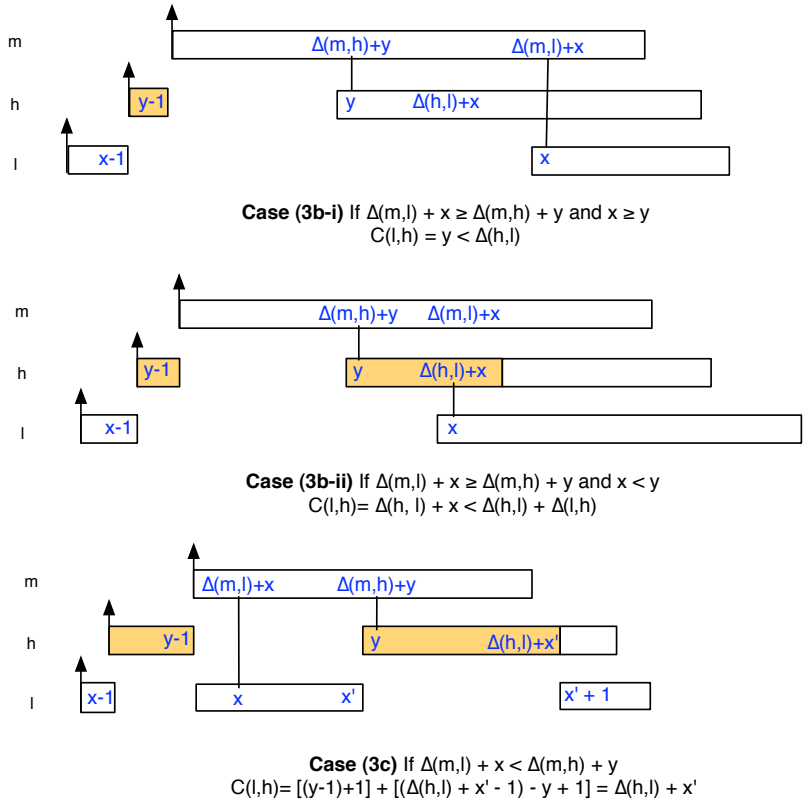


Figure 5.6: Interference of I_h on I_l under multiclass PQS.

i.e., after I_m completes executing $\Delta(m,h)+y$ steps. We note that $x' < \Delta(l,h)$ since otherwise I_l and I_h may be executed concurrently without conflict. In this case, the interference of I_h is $C(l,h) = \Delta(h,l) + x' \leq \Delta(h,l) + \Delta(l,h)$.

From all the above cases, I_h's worst-case interference on I_l is $C_{max}(l,h) = \Delta(h,l) + \Delta(l,h)$. *However, when $L(h) < \Delta(h,l)+\Delta(l,h)$, I_h finishes before I_l reaches $\Delta(h,l)+\Delta(l,h)$; in this case the interference is only $L(h)$. Thus, I_h's worst-case interference on I_l is* $C_{max}(l,h) = \max(\Delta(h,l) + \Delta(l,h), L(h))$.

5.3.3 SQS Multi-class Scheduler

Similar to the single class case when we built the SQS scheduler by modifying the PQS scheduler, the multi-class SQS scheduler is built upon the multi-class version of the PQS scheduler. However, rather than allowing an instance $I_{l,u}$ to steal steal slack from a higher priority instance $I_{h,v}$ if $I_{l,u}$ completed at least $\Delta - S_h$ steps, in the

multi-class case we allow $I_{l,u}$ to steal slack only if it completed at least $\Delta(l,h) - m_l$ steps, where $m_l = \min_{h \in hp(l)} S_h$. This ensures that if $I_{l,u}$ starts stealing slack, then it will always succeed. As it was the case with our previous schedulers, the extension for multiple classes does not increasing the time complexity.

The admission algorithm follows a similar approach to the single-class case. We divide the execution of a query l into a preemptable and a non-preemptable part. The length of the preemptable part is at most $E_l - m_l$, where $E_l = \max_{h \in hp(l)} \Delta(l,h)$ and $m_l = \min_{h \in hp(l)} S_h$. Accordingly, the length of the non-preemptable part is $L_l - (E_l - m_l)$. The worst-case interference of a higher priority instance on a given instance is:

Theorem 9 *An instance I_l may be interfered by a higher priority instance I_h for at most $C_{max} = \min(\Delta(h,l) + \Delta(l,h) - m_l, L_l)$.*

As we proved the worst-case interference for the single class SQS scheduler starting from the PQS interference result, it is straight forward to prove Theorem 9 starting from the results for the interference of multi-class PQS and recognizing that in the case of slack stealing there is a tighter bound on the variable x in Theorem 8 ($x < \Delta(l,h) - m_l$ rather than $x < \Delta(l,h)$).

To compute R'_l we must account for the jitter introduced by slack stealing, i.e., a higher priority instance I_h may delay its start by at most S_h. Accordingly, R' is:

$$R'_l(S_l) = (E_l - m_l) + S_l + \sum_{h \in hp(l)} \left\lceil \frac{R'_l(S_l) + S_h}{P_h} \right\rceil \cdot C_{max}(l,h)$$

where, $\Delta - m_l$ is the maximum length (execution time) of the preemptable part, S_l is the maximum time interval when I_l may be blocked by a lower priority instance due to slack stealing, and $C_{max}(l,h) = \min(\Delta(h,l) + \Delta(l,h) - m_l, L_l)$ is the worst-case interference when slack stealing is used.

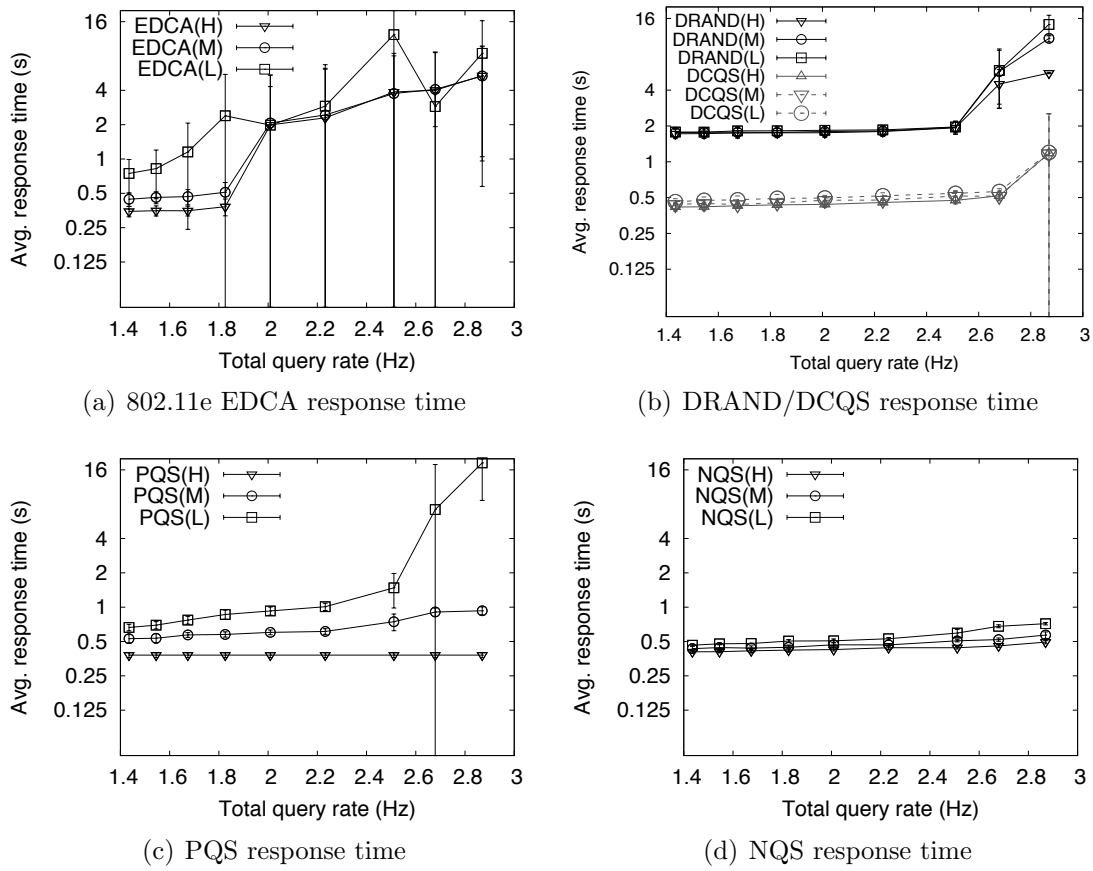
5.4 Handling Packet Loss and Topology Changes

RTQS can be extended to handle both transient and persistent packet loss. To handle persistent packet loss, the routing tree must be changed so as to trigger a recalculation of the plans and minimum step distances. To reduce such reconfiguration overhead, we can modify the routing tree protocol to allow a node to have multiple parents and switch among them in response to packet loss. We modified the planner to construct plans as if a node transmits to all its parents even though in reality it transmits to only one parent at a time. As a result, when a link failure occurs a node may switch to one of its alternative parents without having to recompute any transmission schedules. This approach provides added resilience to topology changes at the cost of lowering the throughput.

RTQS can handle a range of topology changes without recomputing the plans or minimum step distances. Transient packet loss can be handled via Automatic Repeat-reQuest (ARQ) which RTQS supports by increasing the slot size to accommodate multiple transmissions. As in any other TDMA approach, this solution improves reliability at the cost of throughput due to the increased slot size. Note that neither the algorithms nor the analysis needs to be changed for using ARQ.

5.5 Simulations

We implemented RTQS in NS2. Since we are interested in supporting *high data rate* applications such as structural health monitoring we configured our simulator according to the 802.11b settings having a bandwidth of 2Mbps. This is reasonable since several real-world structural health monitoring systems use 802.11b interfaces to meet their bandwidth requirements. An overview of these deployments may be found in [88]. At the physical layer a two-ray propagation model is used. We model interference according to the Signal-to-Interference-plus-Noise-Ratio (SINR) model, according to which a packet is received correctly if its reception strength divided by the sum of the reception strengths of all other concurrent packet transmissions is greater than a threshold (10 dbm in our simulations).



(a) 802.11e EDCA response time

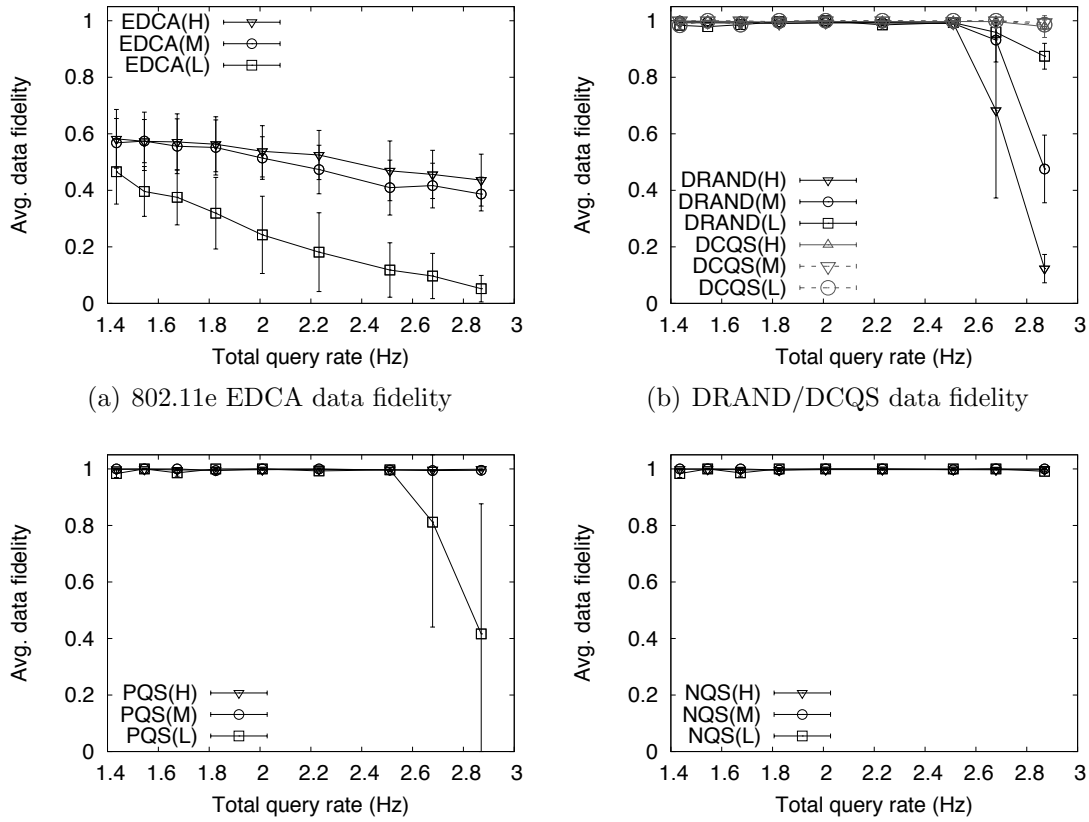
(b) DRAND/DCQS response time

(c) PQS response time

(d) NQS response time

Figure 5.7: Response time of baselines, PQS, and NQS

In the beginning of the simulation, the IC graph is constructed using the method described in [143]. The node closest to the center of the topology is selected as the base station. The base station initiates the construction of the routing tree by flooding setup requests. A node may receive multiple setup requests from different nodes. The node selects as its parent the node that has the best link quality indicator among those with smaller depth than itself. We determined the slot size as follows. We assume that a node samples its accelerometer at 100Hz and buffers 50 16-bit data points before transmitting its data report to its parent. To reduce the number of transmissions, data merging is employed: a node waits to receive the data reports from its children and merges their readings with its own in a single data report which it sends to its parent. In our experiments, the maximum number of descendants of any node is 20, so the maximum size of a data report containing 16-bit measurements is 2KB. Accordingly, we set the slot size to 8.3ms, which is large enough to transmit



(a) 802.11e EDCA data fidelity

(b) DRAND/DCQS data fidelity

(c) PQS data fidelity

(d) NQS data fidelity

Figure 5.8: Data fidelity of baselines, PQS, and NQS

2KB of data. In our simulations, all queries are executed according to the same plan as every node sends its data report in a slot.

For comparison we consider three baselines: 802.11e, DCQS and DRAND[105]. We did not use 802.15.4 as a baseline, since the standard is designed for low data rate applications and hence is unsuitable for our target high data rate applications. 802.11e is a representative contention-based protocol that supports prioritization in wireless networks. In our simulations we use the Enhanced Distributed Channel Access (EDCA) function of 802.11e since it is designed for ad hoc networks. EDCA prioritizes packets using different values for the initial backoff, initial contention window, and maximum contention window of the CSMA/CA protocol. We configured these parameters according to their defaults in 802.11e. We used the 802.11e NS2 module

from [78]. DRAND is a recently proposed TDMA protocol. DCQS is a query scheduling algorithm that constructs TDMA schedules to execute queries. However, neither DCQS nor DRAND support prioritization or real-time transmission scheduling.

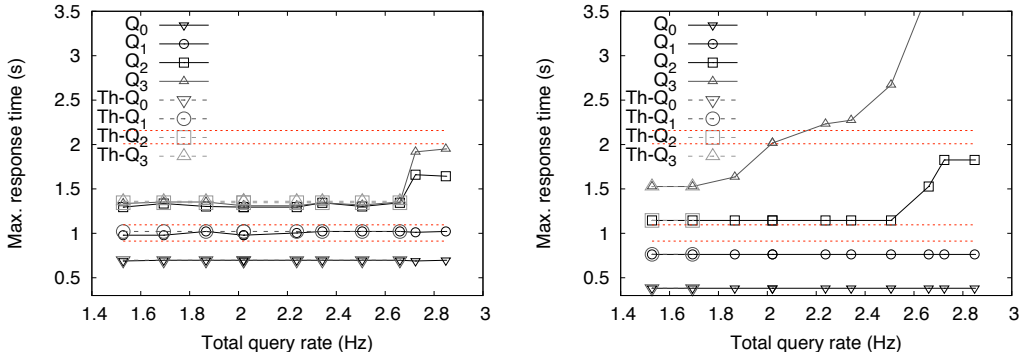
We use *response time* and *data fidelity* to compare the performance of the protocols. The *response time* of a query instance is the time between its release time and completion time, i.e., when the base station receives the last data report for that instance. During the simulations, data reports may be dropped preventing some sources from contributing to the query result. The *data fidelity* of a query instance is the ratio of the number of sources that contributed to the aggregated data reports received by the base station and the total number of sources.

In the following we compare the performance of NQS and PQS with the baselines (see Section 5.5.1) and evaluate the RTQS algorithms under different workloads and validate our response time analysis (see Section 5.5.2).

5.5.1 Comparison with Baselines

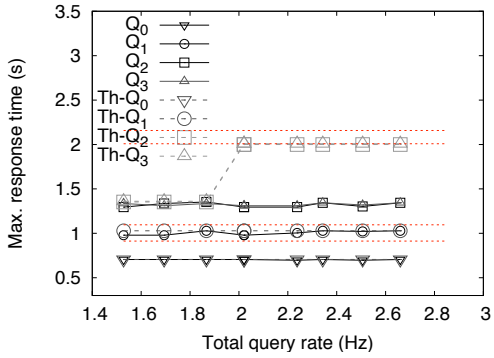
The results presented in this section are the average of five runs on different topologies. The 90% confidence interval of each data point is also presented. All experiments are performed in a 750m ×750m area divided into 75m × 75m grids in which a node is placed at random. We simulate three queries with high, medium and low priorities. The query priorities are determined based on their deadlines: the tighter the deadline, the higher the priority. The ratios of the query periods Q_H:Q_M:Q_L are 1.0:2.2:4.7. The deadlines are equal to the periods.

Figs. 5.7 and 5.8 show the average response time and data fidelity of different protocols as the total query rate is increased from 1.43Hz to 2.87Hz. 802.11e EDCA provides prioritization between queries: when the total query rate is 1.43Hz, the average response times of Q_H and Q_L are 0.34s and 0.74s, respectively (see Figure 5.7(a)). However, 802.11e EDCA has poor data fidelity for all queries (see Figure 5.8(a)). The poor performance of 802.11e EDCA is due to high channel contention, which results in significant packet delays and packet drops. This shows the disadvantage of contention-based protocols for high data rate queries.



(a) NQS max. response time

(b) PQS max. response time

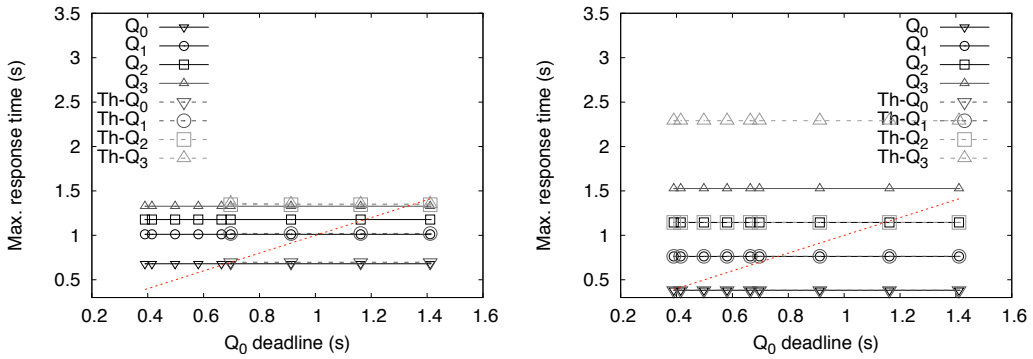


(c) SQS max. response time

Figure 5.9: Response time of queries when workload is varied by changing rates. All queries belong to the same class.

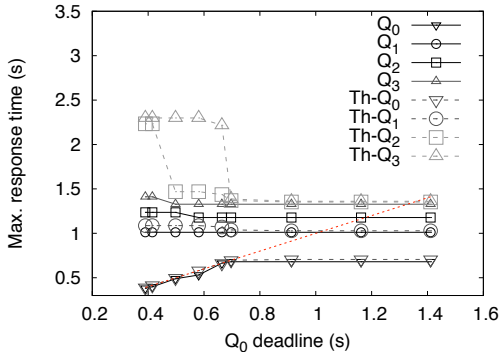
The TDMA protocols, DCQS and DRAND (see Figs. 5.7(b) and 5.8(b)), have significantly higher data fidelity than 802.11e EDCA. The data fidelity results indicate that DCQS provides a higher throughput than DRAND. Moreover, DCQS provides lower response time than DRAND (see Figure 5.7(b)). DCQS performs better because it exploits the inter-node dependencies introduced by queries in WSNs. However, neither protocol provides query prioritization since all queries have similar response times.

In contrast to DCQS and DRAND, PQS provides query prioritization as seen in their response times. For instance, when the total query rate is 2.51Hz, PQS provides an average response time of 0.38s for Q_H, which is 75% lower than the average response time of 1.48s for Q_L (see Figure 5.7(c)). PQS achieves the same query throughput as DRAND, but lower than DCQS due to the high cost of preemption (see Section 5.2.5). PQS achieves close to 100% fidelity when the total query rate is lower than



(a) NQS max. response time

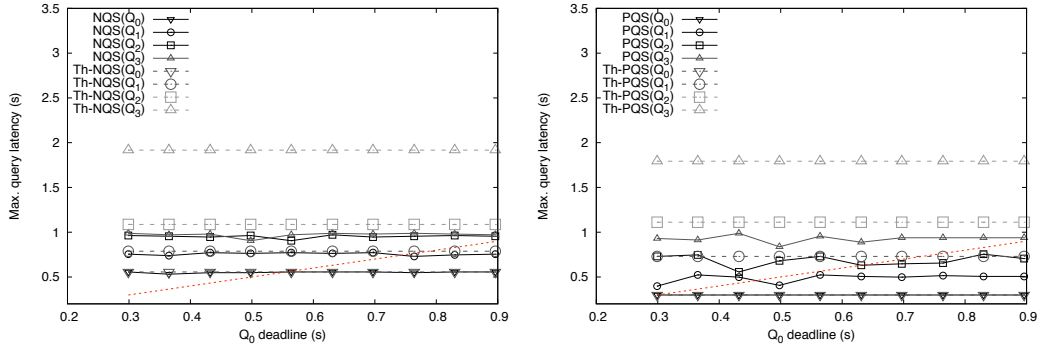
(b) PQS max. response time



(c) SQS max. response time

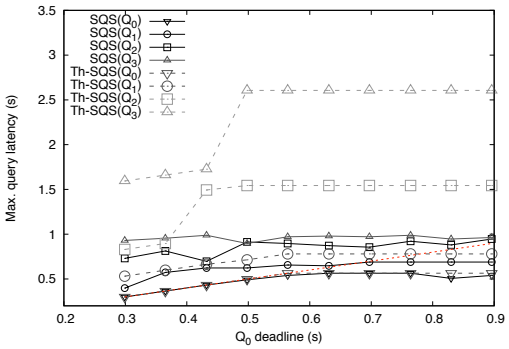
Figure 5.10: Response time of queries when workload is varied by changing the deadline of Q_0. All queries belong to the same class.

2.51Hz (see Figure 5.8(c)). For higher query rates, the fidelity drops because the offered load exceeds PQS's capacity (the schedulability test failed at these rates). NQS also provides query prioritization (the y-axis has a log scale), but the differences in response times are smaller than in PQS due to the priority inversions of non-preemptive scheduling (see Figure 5.7(d)). In contrast to PQS, NQS has close to 100% data fidelity for all queries when the total query rate is as high as 2.87Hz. Therefore, NQS achieves higher throughput than PQS. The comparison of PQS and NQS shows the tradeoff between prioritization and throughput predicted by our analysis.



(a) NQS max. response time

(b) PQS max. response time



(c) SQS max. response time

Figure 5.11: Response time of queries when workload is varied by changing the deadline of Q_0. Experiment includes multiple query classes

5.5.2 Comparison of RTQS Algorithms

In this subsection we compare the performance of all RTQS algorithms and validate their response time analysis. We consider four queries Q_0, Q_1, Q_2, and Q_3 in decreasing order of priority. The ratios of their periods Q_0:Q_1:Q_2:Q_3 is 1.0:1.2:2.2:3.2. In this experiment, we fix the rates of the queries and vary the deadline of the highest priority query.

To evaluate the RTQS algorithms under a broad range of workloads, we perform three experiments. In the first experiment, we fix the deadlines of the queries and vary their rates. In the second experiment, we fix the rates of the queries and vary the deadline of the highest priority query. In the last experiment, we evaluate the performance of the RTQS algorithms for multiple classes.

Experiment 1. Figs. 5.9(a) - 5.9(c) show the measured and the theoretical maximum response times of NQS, PQS, and SQS under different total query rates. The dotted horizontal lines indicate the query deadlines. NQS meets all deadlines when the total query rate is within 2.85Hz. In contrast, PQS supports a lower query rate since Q_3 misses its deadline when the total query rate is 2.23Hz. The long response time of Q_3 is due to the high preemption cost suffered by the low priority queries under PQS. This indicates that PQS is unsuitable for workloads in which the low priority queries have tight deadlines.

Similar to NQS, SQS can support a higher query rate than PQS without missing deadlines. In this experiment, the deadlines are lax and hence preemption is not necessary for meeting them. As such, SQS dynamically avoids preemption and the associated throughput reduction. SQS achieves a slightly lower throughput than NQS because it is limited by the conservative response time analysis. When the admission algorithm decides that the queries are unschedulable, it cannot find a slack assignment for the queries. Therefore we cannot run SQS at a rate beyond its theoretical bound. In contrast, we may increase the rate further under NQS, which achieves a higher throughput than its theoretical bounds because its response time analysis is derived based on worst-case arrival patterns which do not always occur in our simulations.

Experiment 2. In this experiment we increase the deadline of the lowest priority query and vary the deadline of the highest priority query Q_0. This experiment evaluates the RTQS algorithms when the low priority queries have lax deadlines.

Figs. 5.10(a) - 5.10(c) show the maximum response times of NQS, PQS, and SQS, respectively. For clarity, only Q_0's deadline is plotted since in this experiment the other queries always meet their deadlines. PQS meets Q_0's deadline when it is 0.39s. In contrast, NQS meets its deadline only when Q_0's deadline is bigger then 0.69s. NQS misses Q_0's deadline when it is tight due to the priority inversion under non-preemptive scheduling. This indicates that NQS is unsuitable for high priority queries with tight deadlines. Interestingly, under SQS, the response time of Q_0 changes depending on its deadline (Figure 5.10(c)). As the deadline becomes tighter, the response time of Q_0 also decreases and remains below the deadline. We also see an increase in the response times of the lower priority queries as Q_0's deadline is decreased. This is because as Q_0's deadline decreases the lower priority queries may

steal less slack from Q_0. This shows that SQS adapts effectively based on query deadlines. Moreover, note that SQS provides smaller latencies for the lower priority instances than PQS. This is because SQS has a higher throughput than PQS since it uses preemption only when it is necessary for meeting packet deadlines.

Experiment 3. In this experiment, we compare the performance of the RTQS algorithms in the presence of multiple classes. We create different query classes by varying the sources of the queries. For each query class we select at random a fraction of the leaf nodes as data sources. We note that if a node has as descendent a selected leaf node, then it also participates in that query class since it must forward the leaf's data to the base-station. Similar to the previous experiments, data merging is performed as data is routed to the base-station. In this experiment there are two classes: c_0 includes 100% of the leaf nodes while c_1 includes 60% of the leaf nodes. The queries Q_0 and Q_2 belong to class c_0 while Q_1 and Q_3 to class c_1.

Figs. 5.11(a) - 5.11(c) show the maximum response times for NQS, PQS and SQS when the deadline is varied. In each graph we also plot the deadline of Q_0. Similar to the previous experiment, PQS schedules the workload for tighter deadlines of Q_0 than NQS. This is because in contrast to NQS, PQS does not introduce any priority inversions. From the Figs. 5.11(a) and 5.11(b) it is clear that neither algorithm changes its behavior as Q_0's deadline is varied. In contrast, SQS adapts its behavior to meet Q_0's deadline.

In all experiments, the measured response times of all RTQS algorithms are lower than the worst-case response times derived using our analysis. The difference between the simulation results and the theoretical bounds are expected because the analysis is based on worst-case arrival patterns which do not always occur in simulations.

5.6 Summary

High data rate real-time queries are important to many wireless cyber-physical systems. This chapter proposes RTQS, a novel transmission scheduling approach designed real-time queries in WSNs. RTQS bridges the gap between wireless sensor

networks and schedulability analysis techniques which have traditionally been applied to real-time processor scheduling.

We first analyze the inherent tradeoff between throughput and prioritization under conflict-free query scheduling. We then present the design and schedulability analysis of three new real-time scheduling algorithms for prioritized transmission scheduling. NQS achieves high throughput at the cost of priority inversion, while PQS eliminates priority inversion at the cost of query throughput. SQS combines the advantages of NQS and PQS to achieve high query throughput while meeting query deadlines. NS2 simulations demonstrate that both NQS and PQS achieve significantly better real-time performance than representative contention-based and TDMA protocols. Moreover, SQS can maintain desirable real-time performance by adapting to deadlines. Real-time query scheduling provides an promising approach to provide predictable real-time queries for wireless cyber-physical systems.

Chapter 6

Reliability Issues in a Wireless Clinical Monitoring System: A clinical trial

6.1 Introduction

Clinical deterioration in patients in general (non-ICU) hospital units is a major concern for hospitals. Of these patients, 4% – 17% suffer from adverse events such as cardiac or respiratory arrests [14, 128, 32]. A retrospective study found that as many as 70% of such events could have been prevented [79]. A key factor in improving patient outcomes is to detect clinical deterioration early so that clinicians may intervene before a patient's condition worsens. The detection of clinical deterioration is possible because most patients exhibit changes in their vital signs hours prior to an adverse event (median 6.5 hours, range 0 – 432 hours) [16]. Automatic scoring systems aimed at identifying clinical deterioration in patients based on their vital signs are being developed [67, 56]. However, the performance of such systems is significantly affected by having up-to-date vital signs. This may not be a problem in Intensive Care Units where vital signs are monitored by wired monitoring equipment. However, the population that would most benefit from early detection of clinical deterioration is in general or step-down hospital units. In such units, vital signs are often measured manually at long time intervals. For example, in postoperative care, nurses measure the vital signs only 10 times during the first 24 hours following an operation [139]. This could lead to a prolonged delay until clinical deterioration is detected. Thus, it

is necessary to develop a patient monitoring system for collecting the vital signs of patients on general hospital units.

Collecting vital signs in general hospital units poses unique challenges which are poorly addressed by existing commercial telemetry systems. First, for hospitals to deploy monitoring systems in general units they must be *inexpensive*. Existing medical telemetry systems use specialized 802.11 technology and require the deployment of numerous access points connected through a wired backbone. This system architecture results in high equipment and deployment costs making their deployment prohibitive outside specialized units. Second, in contrast to cardiac or epilepsy care which require high data rate EKG or acceleration measurements, the collection of vital signs[10] requires *low data rates*. This creates opportunities to reduce costs by matching hardware capabilities to application requirements: at low data rates, 802.11 may not be the optimal solution in terms of cost and energy consumption. Third, patients in general hospital units may be ambulatory. Hence, it is essential to develop a system which supports *patient mobility*. Moreover, it is unlikely that hospitals will be able to monitor all patients hospitalized in general units. Accordingly, it may be desirable to deploy wireless monitoring systems on a need basis, i.e., when a patient at high risk of clinical deterioration (e.g., who just moved from the ICU to a step-down unit) is admitted to a general hospital unit, the system is *deployed on-demand*. This kind of on-demand deployment is not feasible in existing telemetry systems.

The requirements of low cost and low data rate motivate the development of a patient monitoring system using wireless sensor network (WSN) technology based on the IEEE 802.15.4 standard. While wireless sensor networks as gained attention as a promising technology for elderly care [133], disaster recovery [45], epilepsy care [114], and patient monitoring [33, 86], there has not a in-depth clinical study of the feasibility and reliability wireless clinical monitoring systems for in-patients in eneral hospital units. As a promising step towards real-time clinical detection systems for general hospital units, we present the deployment and empirical study of a wireless clinical monitoring system in a step-down cardiac care unit at Barnes-Jewish Hospital, St. Louis. The developed system monitors the heart rate (HR) and the blood oxygenation (SpO2). Data collected from 32 patients over a total of 31 days of monitoring

[10]The primary vital signs used for patient care in hospitals include temperature, blood pressure, pulse, and respiratory rate, which typically change over minutes.

shows that the median network and sensing reliabilities per patient were 99.92% and 80.55%, respectively. Somewhat surprisingly, the primary source of unreliability was sensing, not networking. While sensing failures occur frequently, the sensors recovered from most of the outages quickly. The distribution of sensing outages is long-tailed containing prolonged outages caused by sensor disconnections. Through trace analysis we show oversampling and automatic disconnection alarms that can substantially enhance sensing reliability with minimum manual intervention. Furthermore, our study indicates the feasibility to detect the clinical deterioration in the two patients who were transfered to the ICU during the trial.

The remainder of the chapter is organized as follows. Section 6.2 presents the related work. The patient monitoring system is described in Section 6.3. The methods and results used during the clinical trial are presented in Section 6.4. Section 6.5 discusses our experience with the design and the operation of the patient monitoring system. Conclusions are presented in Section 6.6.

6.2 Related Work

In this section we review existing medical systems and their empirical evaluation.

Medical Systems: Recently, a number of exciting medical systems have been developed in support of elderly care [133], disaster recovery [45, 33, 68], and patient monitoring [72, 108, 30, 33]. The monitoring of vital signs is a basic function which is supported by these systems. Due to the unique requirements of monitoring patients in general units, we made different design decisions. First, our system design takes advantage of the availability of power in hospital units. This is in contrast to disaster recovery and even in some elderly care settings. Second, some of the existing medical systems support peer-to-peer or publish/subscribe communication [68, 21]. In contrast, we opted for a simpler network architecture in which nodes forward the data to a single base station. Finally, we designed a novel solution for handling patient mobility.

Empirical Evaluations: Numerous patient monitoring systems using cell phones [108, 30], 802.11 [45, 33, 84], and 802.15.4 [133, 45, 26, 86] wireless technologies

have been proposed. The evaluation of these systems typically does not focus on reliability and is usually performed in laboratories at a small scale. In the following, we summarize results obtained from patient monitoring systems deployed in clinical environments.

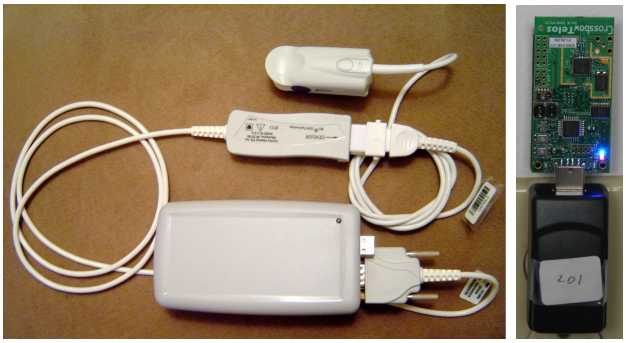
The MEDiSN [72] and SMART [33] projects focus on monitoring patients waiting in emergency rooms. In [72], networking statistics are collected in the emergency room at Johns Hopkins Hospital. The study focuses on understanding the low-level channel characteristics of a typical clinical environment which is particularly useful for developing novel wireless communication protocols. The study focuses on a small scale deployment and, more importantly, it ignores sensing reliability which we show to dominate the overall system reliability. In [33], pulse and oxygenation measurements were collected from 145 patients for an average of 47 minutes (range 5 minutes – 3 hours). No data regarding the reliability of the system is reported. Results from disaster drills are reported in [45, 33]; however, these results do not measure network performance or system reliability. In [26], we presented a preliminary description of our system. The system is evaluated using an indoor testbed and healthy volunteers. In sharp contrast, this chapter provides a detailed description of the system and focuses on its evaluation in a clinical environment. The behavior of patients is known to differs significantly from that of healthy volunteers.

In contrast to prior empirical studies, the study presented in this chapter involves real patients monitored by a large scale system over a long period of time. The patients were monitored in situ to realistically assess the feasibility of wireless sensor network (WSN) technology for patient monitoring. The system we deployed had 18 relay nodes and required multi-hop communication for data delivery. As part of the study, we monitored 32 patients recruited over six months for a total of 31 days of continuous monitoring.

6.3 System

This section presents the system architecture, hardware components, and software we developed for the patient monitoring system. The presentation focuses on the key

design decisions we made to meet the challenges of vital sign monitoring in general hospital units.



(a) Patient node (b) Relay

Figure 6.1: Hardware used for wireless clinical monitoring system

6.3.1 System Architecture

The patient monitoring system has a three tier architecture. The upper tier is formed by a *base station*. The base station runs a data collection application that saves the collected patient data in a local database. In addition, the base station supports remote login for debugging and data backup via an 802.11 link. The lower tier is composed of *patient nodes* (see Figure 6.1(a)). Patient nodes are worn by patients and are capable of measuring their heart rate and blood oxygenation. The middle tier is composed of *relay nodes* (see Figure 6.1(b)). The relay nodes self-organize in a mesh network that provides connectivity between the patient nodes and the base station. The delivery of patient data may involve multiple hops. Moreover, as patients may be ambulatory, we deploy sufficient relay nodes to ensure that a patient node is always one hop away from a relay node.

The system architecture has three features worth highlighting. First, unlike commercial systems, our system does not require the relay nodes to be connected to the hospital's wired network. Table 6.1 shows the price of an 802.11 telemetry system sufficient for monitoring the patients in the step-down unit where the clinical trial was performed. It is worth noting that the access points used by the telemetry system have been modified to better support patient monitoring. The quote was obtained

System	Component	Units	Total price
802.11	Medical grade access points	5	$20,498
telemetry	Ethernet switches	2	$4,046
system	Mounting kits	5	$4,750
	Sensor	20	$8,000
Our system	Infrastructure	18	$1,800

Table 6.1: Prices for a typical 802.11 telemetry system and for the proposed system capable of monitoring the unit part of the trial.

from through the hospital's purchasing department. The equipment cost for our system is also shown. Even though a direct comparison between these figures cannot be made, the significant difference in infrastructure cost gives us confidence that the proposed system is significantly less expensive.

Second, in contrast to other environments in which sensor networks operate (e.g., environmental monitoring), power is widely available in hospitals. We take advantage of this by deploying the relay nodes using USB-to-power adaptors plugged into walls. This simple deployment approach, coupled with the self-organizing features of mesh networking protocols, are the basis for supporting *on-demand deployment*. Note that power management policies are still necessary on patient nodes since they operate on batteries.

Finally, the proposed architecture isolates the impact of patient mobility: mobility may affect only the delivery of packets from the patient node to the first relay, while the remaining hops are over static relay nodes. As discussed in Section 6.3.3, this allows us to reuse the widely used Collection Tree Protocol (CTP) [47] for forwarding data over the static relays and develop a new protocol that finds the best relays to be used by a node even in the case of frequent mobility. Moreover, for similar reasons, we prohibit patient nodes to relay patient data. This has the additional advantage of simplifying the radio power management on sensor nodes.

6.3.2 Hardware

The relay and patient nodes use the TelosB mote as an embedded platform. Each TelosB mote has a 16-bit RISC processor with 48 KB code memory and 10 KB

RAM. Wireless communication is provided using a CC2420 chip which is 802.15.4 compatible. The radio operates in the unlicensed 2.4GHz band and provides a raw bandwidth of 250 kbps. TelosB also has a 1MB external flash which may be used for logging. We opted for the TelosB platform due to its low power consumption and low cost.

A patient node integrates a TelosB mote with a OxiLink pulse-oximeter from Smiths Medical OEM. Both the OxiLink and TelosB support serial communication, albeit at different voltage levels. We developed a custom circuit board which performs the necessary voltage conversions to enable serial communication between them. The circuit also enables the TelosB to turn on and off the OxiLink through a hardware switch controlled by one of the TelosB's I/O pins. This mechanism enabled us to duty-cycle the sensor as discussed in Section 6.3.3. Similar hardware capabilities have been developed and used as part of ALARM-NET [133], MEDiSN [72], AID-IN [45], SMART [33], and WIISARD [68] projects.

6.3.3 Software Components

The patient monitoring system was developed using the TinyOS operating system [54]. The system has three key software components: sensing, networking, and logging. Next, we describe each component.

Network Components

TinyOS supports data collection from nodes through the *Collection Tree Protocol* (CTP). CTP is the *de facto* data collection protocol in sensor networks. CTP has been shown to achieve high reliability in static networks [47]. We developed an initial system prototype which uses CTP to collect data from patient nodes. In this prototype, CTP is deployed both on the patient and on the relay nodes. During the initial testing of the system, we observed that the end-to-end reliability was as low as 82% in the presence of mobility [26].

The following scenario may explain the root cause of the low reliability. The patient node discovers the nodes within its communication range and adds them to its neighbor table. Out of these neighbors, the patient node selects the neighbor with the lowest-cost path to the root as its parent. When the patient moves sufficiently to break the link to the current parent, CTP will select the next lowest-cost neighbor as parent. However, as result of mobility, it is likely that many of the neighbors in the neighbor table are now out of communication range. Accordingly, it is often the case that using the stale information present in the routing table would result in repeatedly selecting nodes outside the communication range of the patient node. Automatic reQuest Retry (ARQ) used by CTP exacerbates this problem by repeating a packet transmission multiple times (e.g., 31 times by default) before dropping the packet and changing the route.

In [26], we validated that CTP's reliability problems were caused by mobility and, as a result, they were confined to first-hop: if a packet reached a relay node, then CTP delivered it to the base station with a relay reliability. Accordingly, a pragmatic approach to ensuring high end-to-end reliability is to isolate the impact of mobility by dividing the problem of data delivery from patients nodes to the base station into two parts: from the patient node to the first relay and from that relay to the base station. We deploy CTP on the relay nodes to forward data to the base station since it achieves high relay over static relay nodes. Next, we designed a companion protocol called *Dynamic Relay Association Protocol* (DRAP) which is deployed on patient nodes to discover and select relays as the patient moves.

The design of DRAP must address three questions: how are neighbors discovered, how to select the best relay to associate with, and how to detect mobility. DRAP discovers new neighbors by listening for beacons periodically broadcast by the relay nodes. DRAP estimates the average Receive Signal Strength Indicator (RSSI) for each neighbor by using a low-pass filter over the RSSI values from both beacons and data packets. DRAP associates with the relay which has the highest RSSI estimate. As packets are sent to the current relay, DRAP keeps track of the number of packet failures. DRAP will invalidate the current neighbor when the number of retransmissions exceeds a threshold. DRAP's approach of combining feedback from the physical (RSSI) and link layer (number of retransmission) in assessing link quality is similar to that proposed in[41]. The novelty of DRAP is that it can also detect mobility by using

a single counter which keeps track of the number of *consecutive* relay invalidations: the counter is incremented when a relay is invalidated and reset to zero when data is successfully delivered to a relay. When the counter exceeds a threshold, DRAP flushes the neighbor table and rediscovers neighbors using its discovery mechanism.

DRAP features a lightweight mechanism for detecting mobility well-suited for the resource constrained devices we are using. We showed that the combination of DRAP and CTP, achieved high reliability even in the presence of mobility. However, the previous results were obtained on a sensor network testbed at Washington University in St. Louis. The results reported the performance of DRAP and CTP over short period of time. In contrast, the results presented in this chapter are obtained from monitoring patients in a step-down hospital unit. A total of 31 days of networking statistics have been gathered during the trial.

The radio may have a significant contribution to the energy budget of patient nodes. In low data rate applications, the radio wastes most of the energy when it is active without transmitting or receiving packets. To address this issue DRAP is augmented with the following power management policy. Typically, power management protocols involve mechanisms that enable a sender and a receiver to coordinate the exchange of packets. These mechanisms assume that power management is performed on both the sender and the receiver. However, in our system, the relay nodes do not require power management since they are plugged into wall outlets. Accordingly, the patient node could turn on the radio when it has a packet to transmit and turn it off after the associated relay acknowledges the reception of the packet. This simple policy handles the bulk of the traffic sent from the patient node to its associated relay without requiring any coordination between them. However, a problem arises during the discovery phase of DRAP: the patient node must be awake to receive beacons from the relay nodes. This problem is solved by keeping the radio awake when the neighbor table is empty (e.g., after it was flushed due to mobility or when a node boots up) for a fixed period of time after the discovery of the first relay node. This allows DRAP to populate its neighbor table with several relays.

This policy has two salient features. First, in contrast to existing power management schemes, DRAP requires neither time synchronization nor additional packet transmissions. Second, the policy is flexible in that the time the radio of a patient node

remains active changes based on the observed link dynamics, variations in workload, and mobility. During the clinical trial we measured the duty cycle of the radio component on several patient nodes. The radio component had a duty cycle between 0.12% – 2.09%. The difference in duty cycles is the result of the DRAP protocol actively changing the associated relays. This is the cumulative result of variations in link quality over time as well as patient mobility.

Sensor Component

The sensor component supports serial communication between the TelosB mote and the OxiLink pulse-oximeter and performs power management. The sensor component measures pulse and oxygenation at user specified rates. Accordingly, every sensing period, the OxiLink sensor is turned on by signaling a hardware switch on the custom board to power up the sensor. The OxiLink sensor provides an indication of the validity of each measurement. The values reported by OxiLink are averages over 8 seconds. As a result, during the first eight seconds after the sensor is powered up, it reports invalid measurements; subsequent measurements may be valid or invalid. Patient movement or improper sensor placement may lead to invalid measurements. The sensor component reads the measurements provided by the OxiLink sensor continuously until a valid reading is received for up to 15 seconds.

Logging Component

We have developed a logging component which is primarily used for debugging and profiling the patient monitoring system. The logging component dedicates a significant portion of the RAM to buffer the generated statistics. Periodically or when the buffer is about to be full, the content of the RAM is saved to the flash in a single batch. We found that batching the flash writing can significantly reduce the amount of time the flash is active, hence reducing energy consumption.

6.4 Clinical Study

To evaluate the feasibility of WSN technology for patient monitoring in step-down or general hospital units, we performed a clinical trial. The trial focuses on answering the following questions:

1. How reliable is the patient monitoring system?
2. What is the distribution of failures for the sensing and networking components?
3. How often nurses need to intervene to achieve high reliability?
4. Does the system provide sufficient resolution for detecting clinical deterioration?

In the subsequent sections, we will answer these questions.

6.4.1 Methods

We deployed the patient monitoring system in a step-down hospital unit at Barnes-Jewish Hospital. We opted to perform the clinical trial in a step-down unit rather than a general unit because patients in step-down units have higher risk of clinical deterioration. Accordingly, there is a higher likelihood that clinical deterioration will be observed during the trial. The step-down unit provides cardiac care for 32 patients and is already equipped with a patient monitoring system.

This study was approved by the IRB of Washington University in St. Louis. Participants were recruited in two phases: the unit's head nurse identified patients which were responsive; we then sought the consent of the identified patients to participate in the trial. On average, one in six patients accepted participation in the trial. The main reason for denying participation was the inconvenience of wearing two monitoring devices: one provided by us and the one already used in the unit. We expect the acceptance rate to be higher on units without telemetry systems.

After obtaining consent, a patient node was placed in a telemetry pouch around the patient's neck. Patients were monitored continuously until their discharge or for up to

Figure 6.2: Deployment at Barnes-Jewish Hospitals. The blue square denotes the base station. Red circles denote relay nodes.

three days. During this time, patients often left the unit for treatment. The nursing staff recorded the times when a patient was not monitored by our system using a time sheet posted in the patient's room. A total of 18 such events were recorded for the 32 participants. This suggests that these events were underreported. The data collected while the patient was not in the unit is excluded from presented results. Upon discharge, the statistics stored in the flash of the patient node was downloaded and stored in the database. This data indicated whether the sensor reported a valid measurement, whether the data was successfully delivered to a relay node, and the duty cycle of the radio, flash, and sensor components. New 9V batteries, monitoring pouches, and disposable pulse-oximetery sensors were used for each patient. After each use, the patient node was disinfected with a concentrated bleach solution.

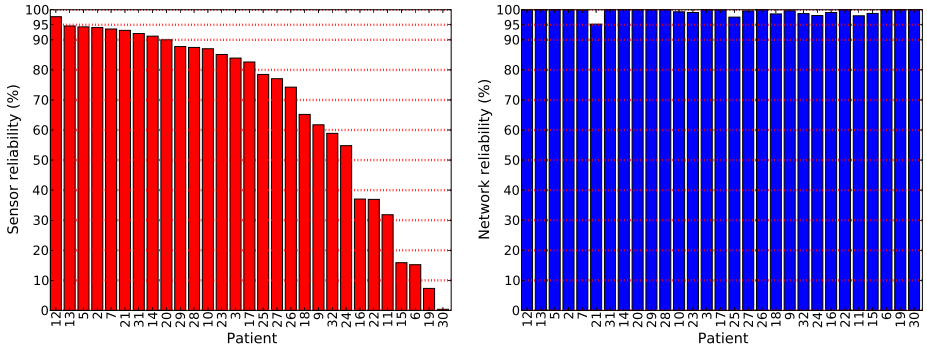
The data collected by the monitoring system was not available to the nursing staff. The hospital was not obliged to act based on the measurements collected by our system. We verified that the measurements taken from patients were valid infrequently (usually daily). If the data provided were invalid, the nursing staff was notified to check if the sensor was disconnected.

The unit has 14 patient rooms and covers an area of 1200 m^2. We deployed 18 relay nodes to provide coverage within the unit as shown in Figure 6.4. Most of the relays were placed in the patient rooms. Hospitals have two independent power circuits: one dedicated for critical equipment and one for non-critical equipment. The relay nodes were plugged into the power outlets on the power circuit dedicated to non-critical equipment. During the trial, the custodial staff unplugged the relay nodes on occasion to power their cleaning equipment. In addition, two relays were destroyed by impact with mobile equipment.. Due to the redundancy of the deployed relays, neither of these events had adverse effects on network reliability. The base station was deployed in a room behind the nurse's station. The base station was powered and access to the hospital's 802.11 wireless network was provided. The system operated on 802.15.4's channel 26 such that it would not interfere with the existing 802.11 network or other telemetry systems. Over the time of deployment, the maximum number of hops varied between 3 – 4.

Patients were enrolled in the study between June 4 and December 4. During this time, a total of 32 patients were enrolled. Demographic data is presented in Table

Variable	Number
Gender	19 male 13 female
Age	average 65 range 34 – 89
Race	17 Caucasian 14 African American 1 undeclared
Adverse events	2 patients transfered to ICU
Total	32
Monitoring time	30 days, 23 hours, 42 minutes

Table 6.2: Study statistics



(a) Sensing reliability (b) Network reliability

Figure 6.3: Network and sensing reliability per patient

6.2. We excluded the results of three patients from the presented statistics. The data from the first patient admitted to the trial was excluded because it had significantly lower network reliability. We determined that an older version of CTP was the source of the problem and updating it to the latest version available solved this isssue. The other two patients were excluded because we collected no data from them. This was the result of a improperly handled exception in the data collection code running on the base station.

The pulse and oxygenation were measured at 30- and 60-second intervals. We selected sampling two rates to gain insight on the impact of sensing rate on sensing reliability and energy consumption. Note that at these rates the resolution provided by our system is orders of magnitude higher than that achieved by manually collecting vital signs. The system collected about 31 days of pulse and oxygenation data. On average,

each patient was monitored for 25.63 hours with a range of 2 – 69 hours. The system most commonly monitored a single patient with up to three patients at a time. During the trial the condition of two patients deteriorated and they were moved to the ICU.

6.4.2 Reliability

In this section, we provide a detailed analysis of the system reliability. To quantify the reliability of the patient monitoring we introduce the following metrics:

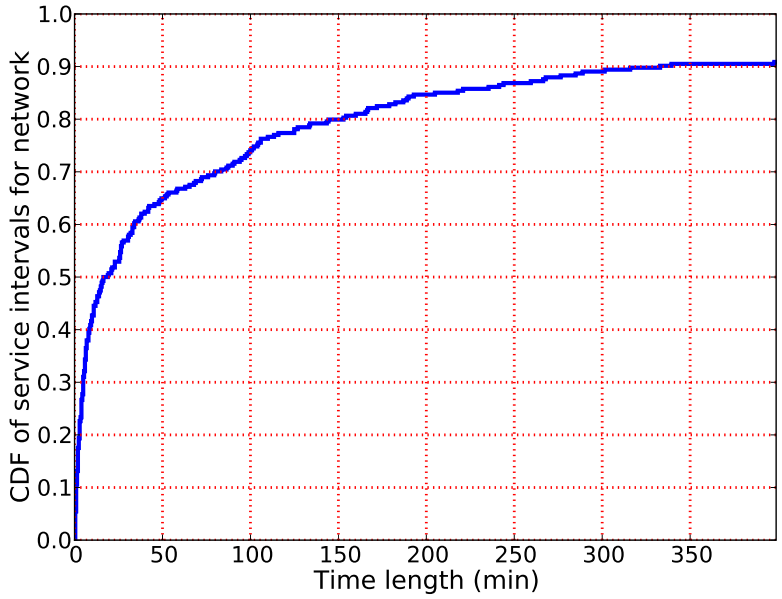
- *Network reliability* is the fraction of packets delivered to the base station.
- *Sensing reliability* is the fraction of valid pulse and oxygenation readings received at the base station. The pulse oximeter provides an indication of the validity of each reading.

To better understand the distribution of failures for the sensing and networking components, it is useful to define *service intervals* and *service outages*. A service interval is a continuous time interval that a component operated without a failure. A network failure refers to the case when a packet is not delivered to the base station, while a sensing failure refers to pulse-oximeter obtaining an invalid measurement. The pulse-oximeter provides an indication of the validity of each reading. A service outage is the time interval from when a failure occurs until a component recovers. The length of *service intervals* is a measure of how frequent failures occur while the length of the *service outages* is a measure of how quickly a component recovers after a failure.

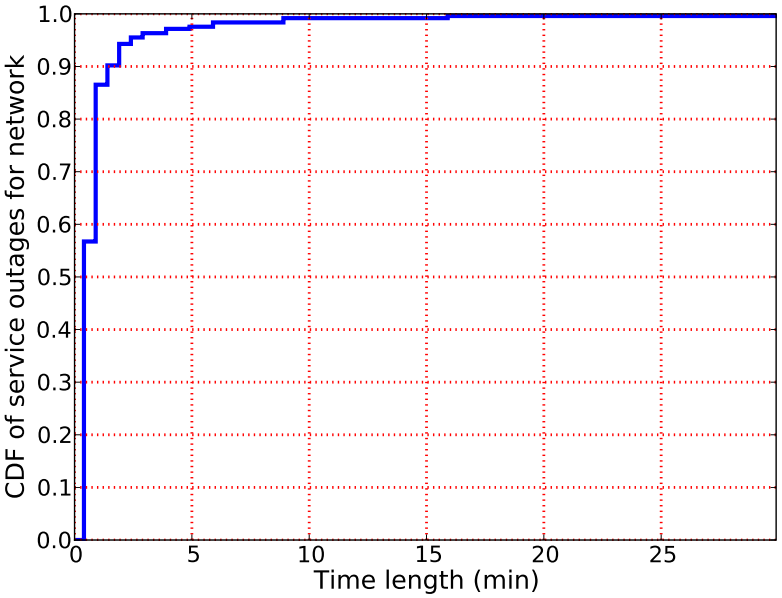
System Reliability

Figure 6.3 plots the network and sensing reliability of each patient. As shown in Figure 6.3(b), the system achieved a median network reliability of 99.92% (range 95.21% – 100%). In contrast, the sensing reliability was significantly lower (see Figure 6.3(a)). The median sensing reliability was 80.55% (range 0.38% – 97.69%).

Several key observations may be drawn from this data. First, the results indicate the system achieved high network reliability for *all* patients in spite of dynamic channel



(a) CDF of service intervals for network



(b) CDF of service outages for network

Figure 6.4: Distribution of service intervals and outages for network component

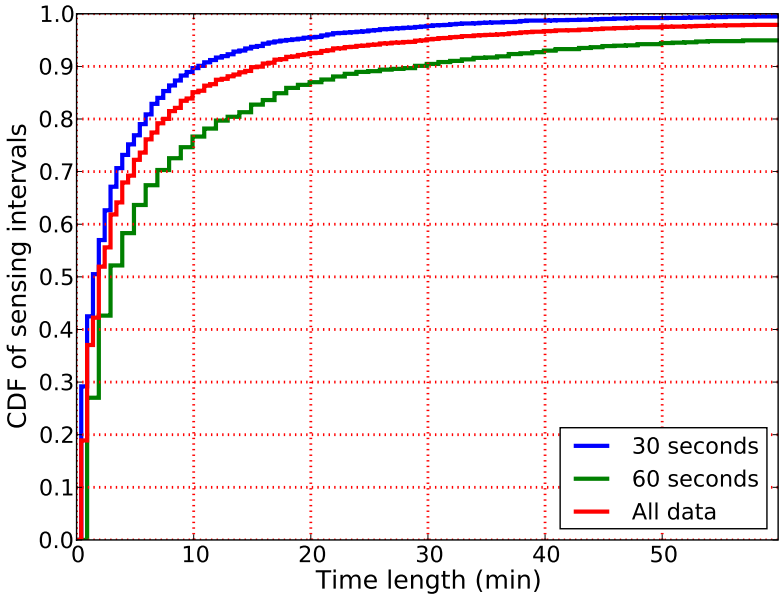
conditions and relay failures. This demonstrates the robustness of CTP and DRAP. Second, the median sensing reliability is sufficient to provide health practitioners with pulse and oxygenation data at two orders of magnitude higher resolution than that achieved through manual collection. However, the wide range of the sensing reliability

is disconcerting: seven patients had reliability below 50%. An in-depth analysis of sensing reliability is deferred to Section 6.4.2. Third, the overall system reliability is dominated by sensing reliability rather than networking reliability. This shows that our future efforts should focus on devising mechanism for improving sensing reliability. Further improvements in networking performance would result in minor improvements in system reliability.
Result: *The overall system reliability is dominated by sensing reliability.*

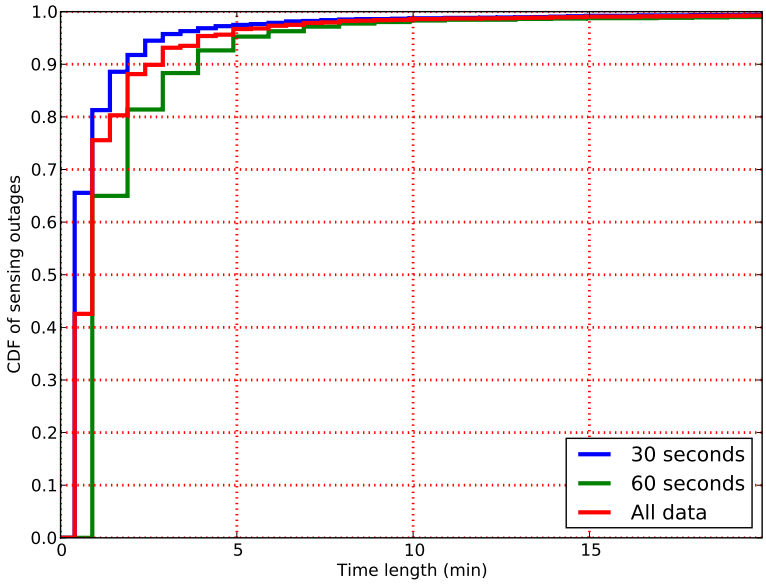
Network Reliability

To analyze the network reliability in greater detail, we consider the distribution of the length of service intervals and outages. Figure 6.4(a) plots the CDF of the service intervals for all patients. The graph shows that the median service interval is 17.7 minutes. Figure 6.4(b) plots the CDF of the service outages for all patients. The graph shows that 80% and 90% of the services outages are less than 0.86 and 1.41 minutes, respectively. Since the measurements are taken every 30 or 60 seconds, we may conclude that it is unlikely to observe more then $2-4$ consecutive packet drops. Thus, the network components recover from failures quickly.
Result: *The network component provides high reliability: networking failures are infrequent and recovery often occurs within a minute.*

We profiled the behavior of DRAP for twelve of the patients. DRAP remained associated with the same relay for five of the patients. This is justified by the low noise level on 802.15.4's channel 26 which does not overlap with other wireless devices. For the remaining seven patient nodes, DRAP changed the relay association at least once. DRAP indicated that mobility was responsible for changes in relay association in four cases. The frequency of mobility was significantly lower than that we previously observed with healthy volunteers [26]. It is also worth mentioning that during the trial a two patients switched rooms. No manual system configuration was necessary for handling this change.



(a) CDF of service intervals for sensor



(b) CDF of service outages for sensor

Figure 6.5: Distribution of service intervals and outages for the sensor component

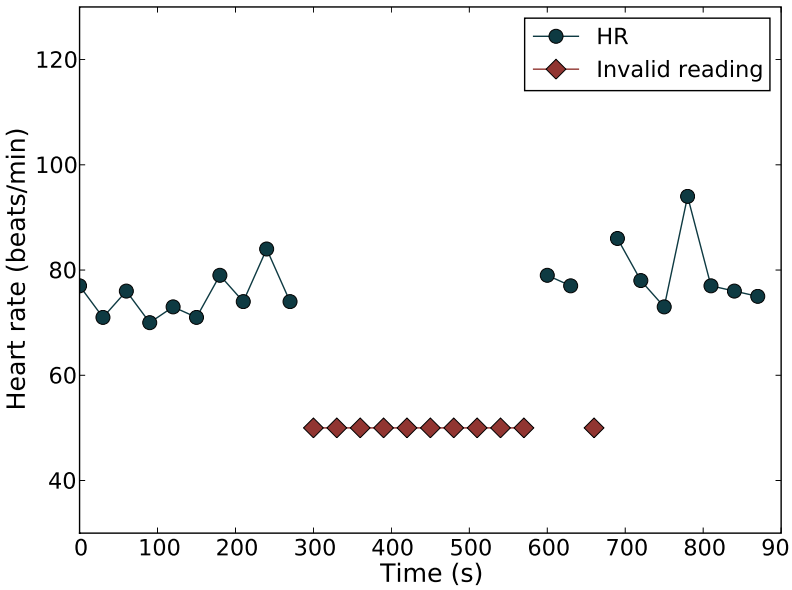


Figure 6.6: Impact of movement on sensing

Sensing Reliability

The quality of pulse and oxygenation readings was significantly affected by patient movement, sensor disconnections, sensor placement, and nail polish; this experience is consistent with results previously reported in literature [116]. Patient movement which includes movement of the arm on which the pulse oximeter was placed, finger tapping, or fidgeting may lead to invalid readings. The impact of patient movement may be significant (see Figure 6.6): when a volunteer moved his hand up and down (300 – 600 seconds), none of the obtained measurements were valid. In contrast, when the patient did not move his arm, a single measurements was invalid. Sensor disconnection also had a significant impact: in 11 of the 32 patients there were sensor disconnections longer than 30 minutes.

The distribution of service intervals and outages for the sensor component is shown in Figure 6.5. We remind the reader that a sensing failure occurs when the pulse oximeter sensor reports an invalid reading. The median service interval is 2.00 minutes, as shown in Figure 6.5(a) when the data from all patients is considered. As few as 8.6% of the service intervals are longer than 17.7 minutes (the mean service interval for the

network component). The short duration of service intervals indicates that sensor failures are common.

Figure 6.5(b) plots the CDF of the duration of service outages. The figure provides two important insights. First, most of the sensing outages are short: 75.2% of the outages last for less than a minute. This suggests that the sensing distribution is characterized by frequent failures which occur in short bursts. These types of failures are the result of patient movement or improper sensor placement. Second, the distribution of service outages is long-tailed: 0.69% of the sensing outages are significantly longer than 20 minutes. The longest service outage lasted 14.3 hours. These long outages are due to sensor disconnections. Nurses did not have access to the patient's data and checked for disconnections infrequently. In section 6.4.3, we consider the effectiveness of an alarm system both in terms of its alarm rates and in on the number of interventions required by the nursing staff.

Result: *The sensor failure distribution is characterized by frequent failures which usually occur in short bursts; disconnections cause prolonged sensing failures.*

Since sensing most failures occur in short bursts, the sensing reliability may be improved through oversampling: the sensor could take measurements at rate higher than the one specified by the doctor. Figures 6.5(a) and 6.5(b) also plot the service intervals and outages when measurements were taken every 30 and 60 seconds. Data indicates that increasing the sampling rate from one sample per minute to two, results in shorter service intervals as well as shorter service outages. The reduction in service outages is expected because the sensor is sampled at a higher rate. The 90-percentile of the service outages is reduced from 3.9 minutes to 1 minutes when the sampling rate is increased from once to twice a minute. The short service outages also explains the increase in the prevalence of short service intervals: since numerous outages are shorter than 30 seconds, then when sensor is sampled at a higher rate, some of the outages may not be observed. The median sensing reliability of the patients monitored at 30 and 60 seconds were 84% and 79%, respectively. This shows that oversampling leads to improved reliability.

Result: *The sensing reliability may be improved through oversampling.*

To further quantify the impact of sampling rate on sensing reliability, we consider the reliability of the system when the requirement of receiving valid pulse and oxygenation

is relaxed to receiving at least one valid reading every 1, 5, 10, and 15 minutes. The updated sensing reliability results are computed based on the collected traces sampled at 30 and 60 seconds. As expected, the sensing reliability per patient increases as the sensing requirement is relaxed, as shown in Figure 6.7(a). In fact, as can be seen in Figure 6.7(b), the increase in sensing reliability can be as much as 62.4%. The patients which benefited most from these improvements had medium and low reliability sensing reliability. Most of the performance improvements were observed when the sensing requirement was increased to 5 minutes; further reductions in the sensing requirement resulted in smaller improvements. This may be explained by the fact that the bursts of sensing errors are short. The highest additional increase in reliability from lowering the sensing requirement from 5 minutes to 10 minutes was 13.4% for patient 16; while the highest additional increase in reliability for lowering the sensing requirement from 10 minutes to 15 minutes was 7% for patient 22. While the sensing reliability of most patients improved, it is worth mentioning that oversampling had no impact on the sensing reliability of eight patients. In the case of these patients, the low reliability was caused by the sensors becoming disconnected rather than intermittent failures. Hence, reducing the sampling requirement had no impact.

6.4.3 Benefits of Disconnection Alarms

As previously discussed, when a sensor became disconnected, the nursing staff should be notified to adjust the sensor. We propose an alarm system to notify the nursing staff when the sensor is disconnected. A disconnection may be detected by keeping track of the time since the the last valid sensor reading was obtained by the sensor. When this time exceeds a disconnection threshold, the alarm is triggered. The selection of the disconnection threshold must consider the trade-off between the nursing effort (i.e., the number of notifications for manual intervention) and the amount of time that no valid sensor readings are obtained. Figure 6.8(a) plots the number of alarms that our system would have triggered for different values of the disconnection threshold based on the data traces collected from the clinical trial. As expected, the system shows that as the disconnection threshold is increased, the number of alarms triggered per day is reduced. When the disconnection threshold is 3 minutes, the number of required interventions per patient is 9. This is comparable to the number

of times pulse and oxygenation are manually measured in postoperative care. A disconnection threshold between 10 – 15 minutes results in less then one intervention per patient per day. At this threshold value, our system significantly reduces the burden on the nursing staff compared to manual collection, which achieving a sampling rate two orders of magnitude higher than manual collection.

Figure 6.8(b) shows the impact of the alarm system on the sensing reliability. The sensing reliability values are computed as follows. Sensing outages longer than the disconnection threshold are identified. The system is penalized for the sensor failures during the time interval from the start of the outage until the disconnection alarm is triggered. The remaining time, from when the disconnection alarm is triggered until the end of the outage, is excluded from the recomputed sensing reliability.

The CDF of patient sensing reliability looks similar for different disconnection thresholds. The most pronounced differences are for patients with reliability in the range 50% – 75%. As expected, the best sensing reliability is obtained when the disconnection threshold is set to its lowest value of 5 minutes, but increasing the threshold interval has only a small impact on sensing reliability. Outside the reliability range 50% – 75%, the impact of the disconnection threshold is negligible. This shows that disconnection thresholds in the range 10 – 15 minutes results in desirable balance between sensing reliability and intervention cost.

Result: *Disconnections may be mitigated through an automatic alarm system with low alarm rates.*

In the following, we estimate the potential benefit of combining oversampling and the disconnection alarm system to achieve even better performance. First, we consider the base case when the sensing requirement is one sample per minute. As previously discussed, reducing the sampling requirement to a sample every 5 minutes results in significant reliability improvements for most patients (see Figure 6.9). Similarly, incorporating an alarm system with disconnection threshold of 15 minutes also results in reliability improvements. Comparing these two curves (*5 min, no alarm* and *1 min, alarm: 15 min*) shows that the two mechanisms act in different ways. The sensor disconnection alarm system has the most impact on patients with low reliability (i.e., those that had disconnections) while the oversampling mechanism

handles intermittent sensing errors. Combining the two mechanisms results in significant improvements: only 3 patients had lower than 80% sensing reliability when the measurements are required once every 5 minutes and a disconnection threshold of 15 minutes is used. From the three patients whose sensing reliability was below 80%, we obtained less than 7 minutes of valid measurements. These makes their reliability unrepresentative for the case when an alarm system would be employed.

Result: *Oversampling and disconnection alarms are complementary and can be combined to achieve further improvement in sensing reliability.*

6.4.4 Detecting Clinical Deterioration

Systems for automatically detecting clinical deterioration may improve patient outcomes by allowing doctors to intervene before a patient's condition worsens. While we have not integrated our system with an automatic scoring system, preliminary results indicate that the developed patient monitoring system provide sufficient resolution to detect clinical deterioration. During the trial, two patients suffered from clinical deterioration and were transfered to the ICU. The pulse and oxygenation data reported by our system are shown in Figure 6.10. Clinical deterioration is visible in patient 3 (see Figure 6.10(a)). Upon being admitted to the unit, the patient had a average heart rate of 55 beats per minute. By the time the patient was transfered to the ICU, the heart rate dropped to 35 beats per minute. A slight degradation in oxygenation is also present. Due to the abrupt deterioration in the patient's condition (about 2 hours), it is likely that his/her vital signs would not have been measured in a unit which does not poses monitoring equipment.

Figure 6.10(b) plots the pulse and oxygenation readings from patient 11. The patient was monitored for 15.4 hours before being transfered to the ICU. During this time, several correlated increases in heart rate and decreases in pulse and oxygenation occurred. In fact, the system provides sufficient resolution to correlate these events such that an automatic clinical deterioration system could have triggered an alarm. These examples highlight that the devised system provides sufficient resolution for analyzing trends in heart rates and pulse oxygenation. As part of our future work, we plan to integrate the patient monitoring system with an automatic scoring system.

Result: *Preliminary results show that the system has sufficient resolution for detecting clinical deterioration.*

6.5 Discussions

Relay Redundancy: The need to ensure network coverage within the step-down unit was one of the concerns raised during the planning of the clinical trial. We considered the possibility of minimizing the number of relay nodes necessary for ensuring coverage. However, this would have required performing in situ measurements to assess the coverage of the relays, which could have been a significant inconvenience to the care providers. Instead, we opted to deploy a redundant network of relays to ensure coverage. The architecture of the system which relies on mesh networking and the availability of power outlets in the hospital makes the deployment of the system effortless. It is worth noting that we were able to redeploy the entire system within 15 minutes. Relay redundancy was essential for tolerating the unplugging of the relays by the cleaning staff and the damaging of relays. Our data indicates that these failures did not impact adversely network performance. Moreover, it is unlike that any packet losses may be attributed to coverage gaps. In retrospect, adopting the more practical solution of deploying additional relay for redundancy was the right choice due the unexpectedly frequent relay failures.

Existing Wi-Fi support: Even though this chapter focuses on reliability concerns, we have not yet discussed the most unreliable part of the system: the 802.11 wireless link from the base station to the hospital's wireless infrastructure. The poor link quality often prevented us from logging into the base station to determine if valid readings were obtained from the monitored patients. Additionally, the transfer of large files was impossible due the same reason. In spite of these issues, we chose not to move the base station in order to maintain a consistent network setup.

It has been argued that a patient monitoring system should take advantage of existing 802.11 infrastructure. If the patient monitoring system would have been required to use this Wi-Fi link, the network reliability would have been significantly lower than that reported in this trial. It is worth noting that the IT department at Barnes-Jewish Hospital invested numerous man-hours to ensure "100% coverage". However, Wi-Fi

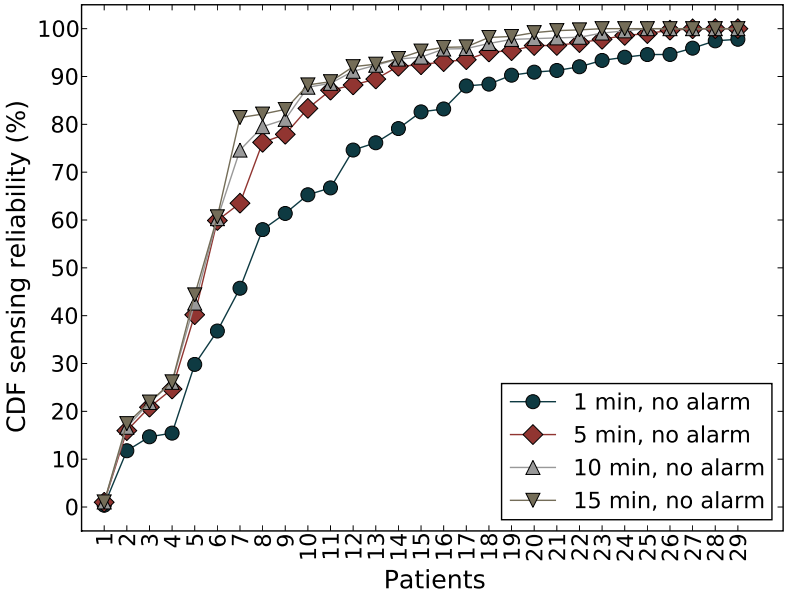
users are accustomed to having to change their location to achieve better performance and, as a result, there is little incentive to deploy more routers to provide true "100% coverage". In contrast, in our system redundancy may be easily achieved and, with 802.15.4 technology, it comes at a low cost.

Power Management: During the clinical trial, patient nodes achieved a life time of up to 69 hours by duty cycling the radio, sensor, and flash. This meets the maximum time we can monitor a patient per the agreement with the Washington University's IRB. The radio and sensor duty cycle was measured on six nodes. The radio consumes 19 mA and had a duty cycle ranging from 0.12% to 2.09%. The sensor draws 24 mA and its duty cycle depends on the sampling rate. Existing pulse-oximeters take up to 8 seconds until average values for hear rate and oxygenation are reported. According, when the sampling rate is 30 seconds, we expect a duty cycle between 26.66% – 50.00%. On the observed devices we obtained duty cycles between 27.3% – 40.27%. Similarly, for a sampling rate of 60 seconds, we expect duty cycles between 13.33% – 25%. In the field, we observed duty cycles in the range 16.24% – 18.97%. These numbers indicate that sensing dominates the energy budget of the patient nodes. The obstacle in achieving lower duty cycles is the prolonged start-up time.

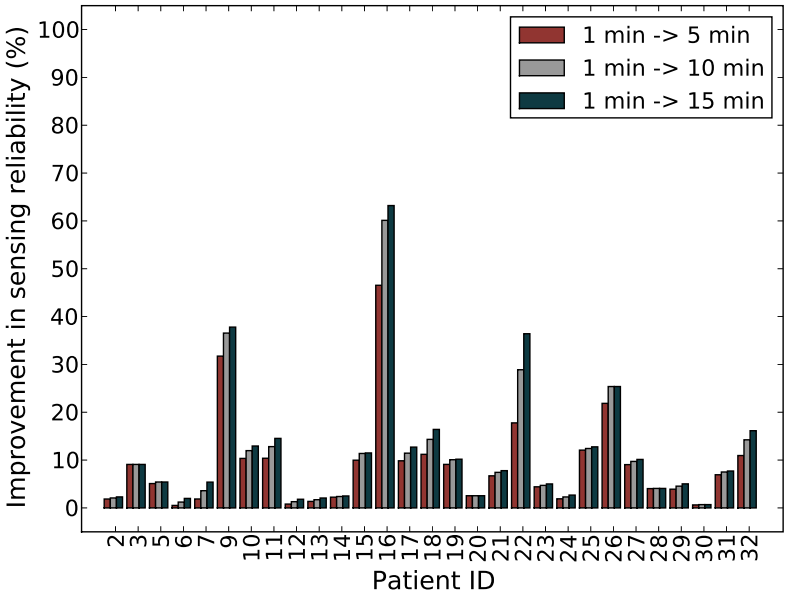
We believe that there are significant opportunities for further reducing the time the sensor is active. For example, a significant amount of energy is wasted when the patient node is left active while a patient goes for treatment outside the unit. A simple policy of reducing the sampling rate after multiple consecutive sensing failures could save significant energy. However, note that even without any of these more complex power management policies, we achieved a lifetime of 3 days. Interesting opportunities also exist for improving energy efficiency by using additional sensors. For example, accelerometers which have lower energy consumption than pulse oximeters, may be used to asses if a patient is moving. The detection of patient movement would prevent us from turning on the pulse oximeter sensor when it cannot provide valid readings and waste energy as a result. The cessation of patient movement would constitute a trigger for the start of measurements.

6.6 Summary

This chapter presents the design, deployment, and evaluation of a wireless pulse-oximetry monitoring system in a hospital unit. the study presented in this chapter involves real patients monitored by a large scale system over a long period of time. The patients were monitored in situ to realistically assess the feasibility of WSN technology for patient monitoring. The system we deployed had 18 relay nodes and required multi-hop communication for data delivery. As part of the study, we monitored 32 patients recruited over six months for a total of 31 days of continuous monitoring. Our work made several main contributions to wireless sensor network technology and clinical monitoring. (1) Our network achieved a 99.92% median reliability over 31 hours of monitoring. The high network reliability indicates the feasibility of applying wireless sensor network technology for clinical monitoring and the efficacy of separating end-to-end routing from first-hop relay association in a clinical environments. (2) System reliability is dominated by the sensing reliability of the commercial pulse oximeter. Sensing failures are frequent, but usually occur in short bursts with the exception of prolonged sensor disconnections. Oversampling and disconnection alarms that can substantially enhance sensing reliability. (3) Our study provides clinical examples that show the potential of wireless clinical monitoring system in enabling real-time detection of clinical deterioration in patients. A promising step towards real-time clinical detection systems for general hospital units, our work also points to several important future areas of research, such as the integration of real-time clinical monitoring systems with the electronic health record systems and the development of clinical event detection algorithms based on real-time sensor streams.

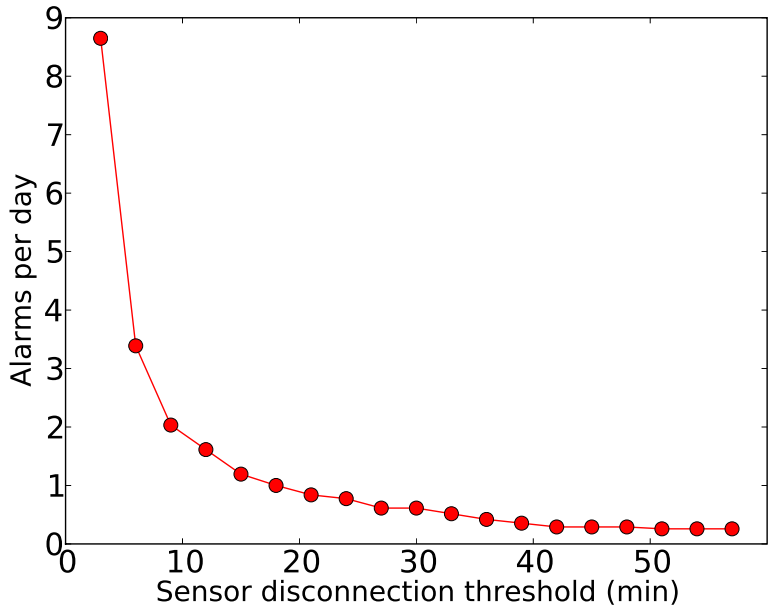


(a) Impact of oversampling

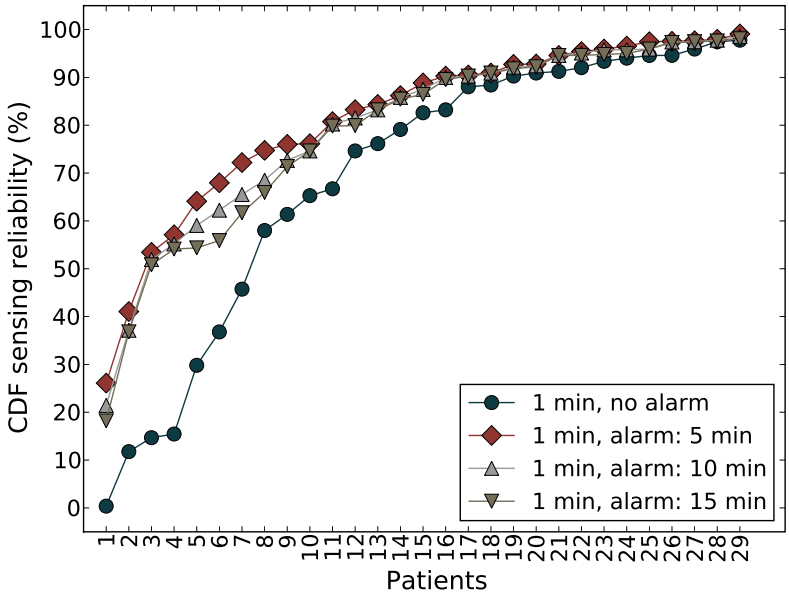


(b) Improvement in reliability

Figure 6.7: Impact of oversampling on sensing reliability



(a) Number of interventions



(b) Impact of alarms on sensing reliability

Figure 6.8: Expected performance of a sensor disconnection alarm system

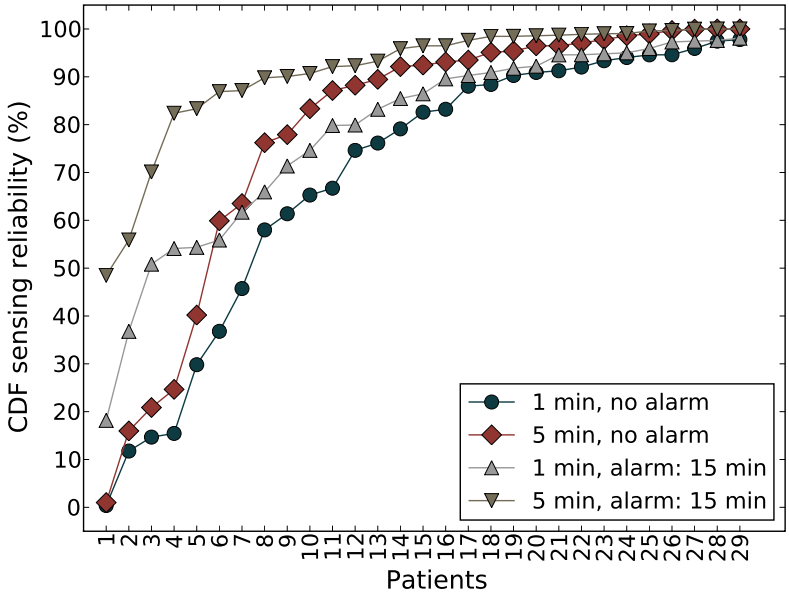
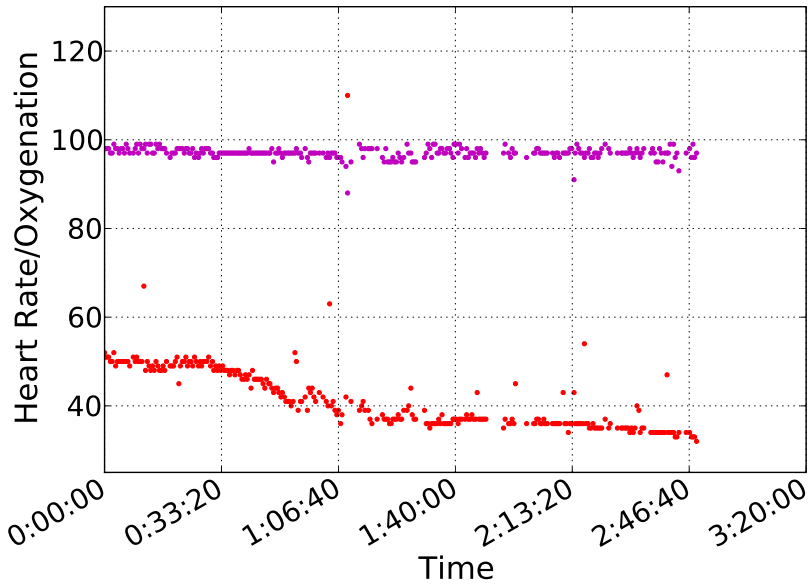
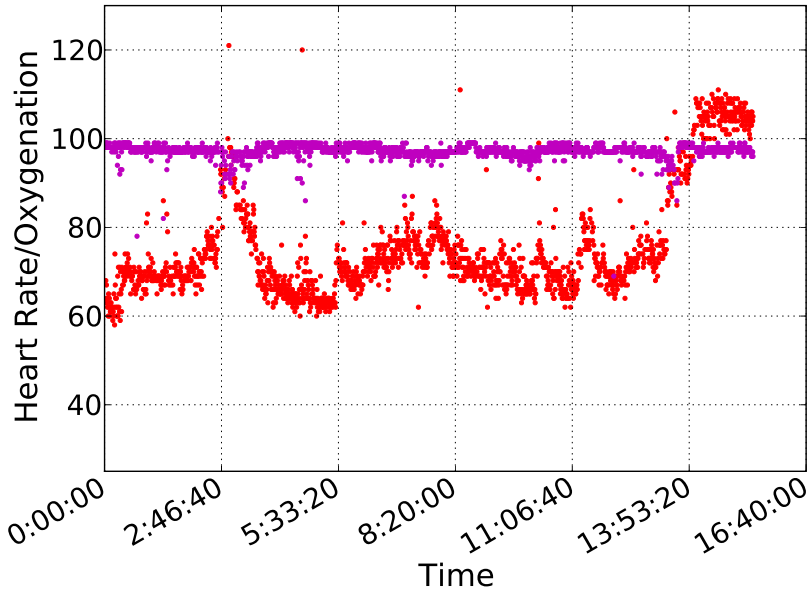


Figure 6.9: Combining oversampling and sensor disconnection alarm systems



(a) Patient 3



(b) Patient 11

Figure 6.10: Pulse (red) and oxygenation (purple) measurements from patients which suffered clinical deterioration

Chapter 7

Practical Modeling and Prediction of Radio Coverage in Indoor Sensor Networks

7.1 Introduction

Sensor network applications involving mobile entities commonly require the deployment of wireless networks that cover a physical region. Examples of such applications include elderly care [132] and patient tracking and monitoring [36]. Our interest in this topic is motivated by a medical application which involves the collection of pulse and oxygenation readings from patients in general hospital units. Unlike patients in the intensive care units, the patients in general hospital units are often ambulatory. To support patient mobility, our system [26] requires enough relay nodes so that there is always at least one link from the patient to some relay. During the deployment of the system at Barnes Jewish Hospital, we became acutely aware of the lack of tools which would enable system managers to effectively assess the coverage of a deployed network. More specifically, we are interested in determining the reception coverage of a relay: i.e., the set of points (x, y) where a node would be able to transmit a packet to at least one relay with a PRR above a user-specified threshold[11].

[11]The techniques proposed in this chapter are also applicable to the network's transmission coverage: i.e., the set of points that can receive transmissions from at least one relay. We focus on reception coverage in this chapter, since our target application entails data collection. Henceforth, we use the term "coverage" to mean "reception coverage".

The current best practice for assessing network coverage is to exhaustively measure link quality at numerous locations with deployed relays. This process is labor intensive and leads to significant deployment costs. Worse, physical changes (e.g., reconfiguring cubicles) or changes in the radio properties (e.g., switching radio frequency due to interference) may invalidate these measurements, leading to significant maintenance costs.

What is needed is a tool which can assess the coverage of a wireless network without an exhaustive survey. The key to assessing wireless coverage lies in effectively modeling radio propagation in the deployment environment, including obstacles that can attenuate the radio signal. Within the 802.11 networking community, there are a handful of tools which use ray tracing techniques to model signal propagation [42]. These tools require precise characterizations of the location and radio properties of objects that can significantly affect radio propagation, such as walls, bookshelves, or filing cabinets. In many indoor environments, such as office environments, these obstacles are numerous; for example, our $1977m^2$ indoor testbed contains 239 walls. Measuring each wall directly would impose an excessive burden on the user.

A less labor-intensive approach is to collect a set of link quality measurements from the environment. This training data can then be fit to some radio propagation models in order to estimate the value of each parameter. Indeed, this approach has proven effective in outdoor environments [107]. However, our empirical study shows that this approach is unsuitable for complex indoor environments. This occurs because obstacles, antenna orientation, and distance between sender and receiver affect signal propagation to different degrees indoors and outdoors. While complex models may be constructed to account for all these factors, there is an important tradeoff between model complexity and measurement effort: as radio models are made more complex, more data is necessary to accurately fit the additional parameters. Thus, in this chapter, we consider the problem of how to effectively predict radio coverage in complex indoor environments from a *small* set of training data.

An empirical study in two office buildings shows that the best tradeoff between model realism and model complexity lies in automatically classifying obstacles into groups with similar attenuation. Using this knowledge, we divide the problem into two parts. We first predict the receive signal strength (RSS) at the relay from any point on the

floor plan. Then, based on the RSS predictions and an RSS threshold for predicting good-quality links, we determine each relay's coverage.

This chapter makes the following key contributions. First, we present an in-depth empirical study that characterizes the accuracy of RSS predictions based on several propagation models. The study shows the relative importance of modeling various aspects of wireless propagation such as antenna orientation, wall attenuation, and distance between sender and receiver. More importantly, the study shows that complex models do *not* necessarily produce accurate estimates of signal strength: the best performance is achieved by a family of models which classify walls based on attenuation into a small number of groups. We also propose an automatic process for selecting the best such model for the provided amount of training data, reducing errors by up to 9.7% compared to the classical log-normal radio propagation model [12]. Next, we develop a practical *Radio Mapping Tool* (RMT) which predicts the coverage of one or more relays. As a key component of RMT, we develop a novel automated wall classification algorithm to be used with the chosen radio model. We then characterize the accuracy of this tool in two different buildings with differing construction properties. We find that the combination of our chosen radio model with our wall classification scheme reduces the false positive rate (i.e., predicted coverage where the ground truth indicates otherwise) by as much as 54% compared to the log-normal model, based on a sampling density of only 0.01 samples/m^2.

The remainder of the chapter is organized as follows. In Section 7.2, we discuss existing studies on characterizing wireless signal propagation. In Section 7.3, we overview several established radio models and discuss their applicability to indoor environments. In Section 7.4, we discuss methods to classify walls, including a computationally efficient algorithm that automatically performs this classification. The RSS prediction accuracy of different propagation models is assessed in Section 7.5. In Section 7.6, we present a radio mapping tool built based on the insights gained from our empirical study. Section 7.7 evaluates the efficacy of our radio mapping tool through a case study. We then conclude in Section 7.8.

7.2 Related Work

A key challenge in modeling radio properties is that low-power wireless links have complex, often probabilistic properties [145, 104, 121, 119, 83, 44]. The classical *log-normal* model [148, 122] models a node's transmission strength and signal decay over distance. As we show in Sections 7.5 and 7.7, the log-normal model is overly simplistic, resulting in significant prediction errors.

A deficiency of the log-normal model is that it does not capture the non-isotropic antenna pattern observed even with "omnidirectional" antennas [119, 144, 107]. [144] demonstrates that these non-regular radiation patterns can have a significant effect on routing performance in an outdoor wireless sensor network. [107] shows a similar effect for two outdoor Wi-Fi mesh networks. Both studies propose a *sectorization* approach that divides each node's signal into sectors, then attempts to independently model the signal properties of each sector. Our own study finds that the non-isotropy of antenna patterns is also important in indoor environments. However, we also find that indoors this effect is less significant then the attenuation caused by obstacles.

[107] expands the sectorization model to explicitly model non-isotropic antenna patterns. Exhaustive link data is collected at various points around each feature to individually estimate its attenuation. Our own study shows that modeling obstacle attenuation can also significantly improve coverage prediction indoors. However, our work differs from [107] in two key ways. First, as discussed above, we do not model antenna patterns; the impact of obstacles are more important in an indoor environment, and modeling non-isotropy introduces a large number of parameters that are difficult to estimate from a small number of samples. Second, [107] directly measures architectural features, which is impractical and labor-intensive in typical indoor environments such as offices, assisted living facilities, and hospitals. A novel feature of our work is that we leverage the fact that the walls in any given building can be classified into relatively few classes of similar attenuation, greatly reducing the amount of data needed to adequately estimate their attenuation. Moreover, we propose an algorithm which automatically classifies walls using a small set of training data, without requiring architectural knowledge or direct measurements of each wall.

At the other end of the complexity spectrum, researchers have proposed site-specific techniques involving ray tracing [97, 110]. [42] presents a tool for predicting signal strength of 802.11 access points at different locations. A fundamental limitation of these techniques is that they rely on the user to provide locations and attenuation coefficients for each partition or obstacle. Tables which provide the attenuation of different wall types [112] can alleviate this burden somewhat, though this still requires knowledge of the building's construction materials and may not capture the effect of objects like metal bookshelves that can alter a wall's attenuation. In contrast, our approach automatically estimates the attenuation of walls from training data.

Also closely related to our work are two recent chapters which look at sensing coverage. [74] proposes a framework which uses Gaussian processes to model sensing and communication costs. A disadvantage of Gaussian processes is that they cannot effectively model discontinuities such as those observed when a signal passes through walls. In contrast, our approach explicitly models wall attenuation, which our study in Section 7.5.5 shows to be significant. [58] proposes a method for determining a sensor's sensing radio range through hierarchical sampling. This approach is complementary to our own, since it deals with efficient sampling strategies for refining coverage boundaries; our work focuses on processing the collected samples to predict coverage.

7.3 Radio Propagation Models

Propagation models optimized for different wireless technologies and environments have been proposed in literature [50, 6]. We assume that nodes operate on a fixed frequency and transmission power. The models presented in this section focus on three characteristics which may significantly affect signal propagation in indoor environments: (1) the distance between the sender and receiver, (2) antenna orientation, and (3) the impact of walls. We note that the models considered in this section do not model multi-path propagation. While multi-path propagation may be modeled through ray-tracing techniques, such approaches are usually computationally demanding and require a precise characterization of the environment. Since our goal is to develop an interactive radio mapping tool, we opted to ignore these effects.

Moreover, we show that the simple models proposed in this section may accurately predict coverage.

By their nature, these models' parameters are estimated from an (ideally small) set of training data. The need for training data represents an important trade-off that we will revisit throughout this chapter: while an overly simplistic radio model may not provide an accurate estimate of communication coverage, adding more complexity will not *necessarily* improve the model's performance. As more parameters are added, more training data is needed to adequately estimate them — conversely, for a fixed training data size, the estimates for each parameter may degrade as more are added.

Thus, the challenge in creating a realistic radio model lies not only in identifying what factors can affect signal propagation, but also *which* of these factors are the most important to capture. Our goal is to identify the model with the best trade-off between prediction accuracy and the number of samples needed to estimate its parameters. This model will ultimately be used in our Radio Mapping Tool to generate signal strength predictions. The models presented in this section will be evaluated empirically in Section 7.5.

Log-Normal Shadowing: Under the log-normal model, signal strength decays exponentially as a function of distance. Let $d(s, r)$ be the distance between the sender node s and the receiver node r. The receive signal strength $P_r(s, r)$ at r from a sender s is given by: [12]

$$P_r(s, r) = \alpha - 10\beta \log_{10} d(s, r) + \sigma \tag{7.1}$$

Here, α represents the transmission power at a reference distance of 1m and β represents the pass loss exponent. σ models shadowing (i.e., the random signal variations between sender and receiver) and is usually considered to be a normally distributed random variable.

Prior empirical studies have shown that this model may accurately predict the receive signal strength of low-power radios in outdoor environments [83] and in indoor environments where nodes have line-of-sight [83, 148]. However, this model does not account for the impact of walls, which commonly have major impacts on the coverage of sensor networks deployed indoors.

Sector-Based: Prior literature has extended the basic log-normal model to capture the fact that many low-power radios have non-isotropic radiation patterns [145]. That is, even when nodes are positioned at equal distances from the sender, they may observe significantly different receive signal strengths.

The receive signal strengths depend on the relative orientation of the sender and receiver. However, to simplify the problem, the relative position of the sender and receiver is commonly kept constant during data collection. In this case, non-isotropic behavior is accounted for by parametrizing α by the angle θ between the line connecting s and r and a fixed frame of reference:

$$P_r(s,r) = \alpha(s,\theta) - 10\beta \log_{10} d(s,r) + \sigma \tag{7.2}$$

$\alpha(\theta)$ may be a non-linear function [145]. As a result, non-linear optimization techniques would be necessary for fitting the model. To simplify fitting, the impact of antenna orientation may be captured by discretizing θ into a number of sectors. This enables us to use linear fitting to estimate all model parameters.

Per-Wall Attenuation: In indoor environments, walls may significantly attenuate wireless links. Hence, incorporating walls into the radio propagation model can improve its signal strength predictions.

An intuitive way of modeling wall attenuation is to assume that each wall $w_i \in W$ in the environment attenuates the signal by a constant factor γ_{w_i}. If we let $I_{s,r}$ be the set of all walls which intersect a virtual line between s and r, then the signal strength at r is:

$$P_r(s,r) = \alpha - 10\beta \log_{10} d(s,r) + \sum_{w \in I_{s,r}} \gamma_w \tag{7.3}$$

This model may also be modified to incorporate non-isotropic radio range by treating α as a function of θ as previously discussed.

Several measurements should be taken through each wall to accurately estimate γ. This may be a significant burden in some environments; for example, one building in our environment contained 128 walls in $1020m^2$ of floor space.

Log-Normal	Sector-Based	Per-Wall	Wall-Class
2	$NS * n + 1$	$\|W\| + 2$	$\|C\| + 2$

Table 7.1: Parameters per model

Wall-Class Attenuation: A pragmatic alternative to the per-wall scheme is to group walls into a few classes, reflecting the fact that only a few types of walls are used in construction. For example, the building shown in Figure 7.2 mainly uses two kinds of walls: cinder block and drywall. Given a set of classes C, a mapping $\Pi : W \to C$, and an attenuation coefficient Γ_{c_i} for each class $c_i \in C$, the signal strength at a node r is:

$$P_r(s,r) = \alpha - 10\beta \log_{10} d(s,r) + \sum_{w \in I_{s,r}} \Gamma_{\Pi(w_i)} \tag{7.4}$$

Table 7.1 summarizes the number of parameters used by each model. As later highlighted by the empirical results presented in Section 7.5, one of the key challenges of Radio Mapping is selecting a model which achieves the best prediction accuracy given a number of measurements. Models with small number of parameters have the advantage of requiring a small number of measurements for determining their parameters. Moreover, when only a limited number of parameters are available, it is imperative to focus on the factors which have the most significant impact on signal propagation.

The log-normal model has only two parameters which need to be evaluated. In contrast, the sector-based model has as many as $NS * n + 1$ parameters, where NS and n are the number of sectors and relays, respectively. The two buildings used in our experiments contained 64 nodes and 28 nodes, respectively. Due to the multiplication between number of sectors and number of relays, such environments will generate models with numerous parameters. Similarly, the per-wall model accounts for the attenuation of walls; as a result, one expects it to be more accurate. The number of parameters used by this model is $|W| + 2$. $|W|$ may be as high as 100 in typical office buildings, resulting in a model with a significant number of parameters. Therefore, we expect the per-wall model to require copious measurements as training data. Moreover, obtaining good statistics for the attenuation of a wall potentially requires multiple measurements per wall. We hypothesize here (and show in Section

7.5) that models with numerous parameters require a significant amount of training data, making them impractical for our Radio Mapping Tool.

In contrast with the previously discussed models, the wall class model requires $|C|$ parameters. In our experience, typical values for $|C|$ are between 1– 5 wall classes, significantly reducing the number of model parameters. Such a model is particularly attractive for our Radio Mapping Tool since it would require only a small number of measurements. However, it also creates a new problem: a mapping Π from walls to classes needs to be constructed. In the next section, we will present an efficient algorithm for constructing this mapping.

7.4 Automatic Wall Classification

One way to construct this wall classification is to manually classify walls based on their construction material. Linear regression may then be used to fit the remainder of the model's parameters as described above. However, manual wall classification is labor-intensive and requires architectural information that may not be readily available to application developers or network managers.

Hence, we propose to classify each wall automatically. The problem of automatically classifying walls into classes may be addressed in the Expectation Maximization (EM) framework. The EM framework is best suited for finding the maximum likelihood estimate when the model depends on latent variables, which in our case are the wall classes. We propose the novel application of the EM framework to automatically classifying walls.

The input to the classification algorithm is based on link statistics collected by the user when located at a measurement location. Multiple packets are broadcast at each measurement location and the relay nodes record their RSS. For each link formed between a relay and a node positioned at a measurement location, we provide the median RSS as vector y and the Euclidean distance between the link's endpoints as vector d. The set of walls and wall classes are provided as W and C, respectively.

$[\alpha, \beta, \Gamma, \Pi]$ = compute-parameters(y, d, W, C):
1: improvement = true;
2: for each wall $w \in W$:
3: $\Pi(w) = rand(C)$;
4: while (improvement):
5: improvement = false;
6: $[\alpha, \beta, \Gamma] = regress(y, [d; \Pi])$;
7 : for each wall $w \in W$ in random order:
8: $\Pi_{new} = \Pi$ and $c_{old} = \Pi(w)$;
9: for each class $c \in C$:
10: $\Pi_{new}(w) = c$;
11: $\hat{y} = \alpha(s) - 10\beta \log_{10} d(s, r) + \sum_{w \in I_{s,r}} \Gamma_{\Pi_{new}(w)}$;
12: $SSE(c) = \sum_{i=1}^{|y|}(y(i) - \hat{y}(i))$;
13: $c_{best} = arg\, min_c SSE(c)$;
14: if $(c_{old} \neq c_{best})$:
15: $\Pi(w) = c_{best}$;
16: improvement = true;
17: break;

Figure 7.1: Wall classification algorithm

Figure 7.1 presents the pseudocode of this algorithm. Initially, each wall is assigned to a random class. The algorithm then proceeds in two stages, repeating until changes in wall classification stop improving the sum of squared errors (SSE) between the predicted signal strengths ($\hat{y}$) and the actual signal strengths (y). In the first stage (line 6), the algorithm uses linear regression to fit the parameters α and β, as well as the attenuation coefficient Γ for each wall class. The second stage (lines 7–16) aims to improve the mapping of walls to classes with these values of α, β, and Γ fixed. This is done by considering each wall w in random order, computing the SSE when w is assigned to each class in C. If reassigning w results in a smaller SSE, then w's classification is updated accordingly and the algorithm goes back to executing the first stage with an improved wall classification. Otherwise, the algorithm considers the next wall. The algorithm terminates when no wall may be assigned to a new class that reduces the SSE. The values of the parameters α, β, and Γ are then returned along with the mapping Π of walls to classes.

This algorithm has two noteworthy features. First, it is much less computationally expensive that an exhaustive search. The wall-reassignment stage considers at most

$|C| \times |W|$ potential assignments at each iteration. Thus, in practice, this algorithm could be executed in under two minutes on a modern laptop PC even when predicting coverage of relays spanning an entire building (tens of relay locations and about a hundred relay locations).

Second, the algorithm is guaranteed to converge. This is because the algorithm reduces the squared error at each step until it terminates. There is no guarantee on the optimality of the solution, since it may get stuck in a local minimum. Because of the random initial assignment of walls to classes and the random ordering in which walls are reclassified, the algorithm may return different values each time it is run. Accordingly, we may further improve the squared error by repeating the algorithm several times and returning the parameters which resulted in the lowest squared error.

7.5 Empirical Model Comparison

In this section, we present an empirical study which aims to address three questions at the core of our Radio Mapping technique: (1) which factors affect signal propagation in an indoor environment, (2) how model accuracy affects the number of samples needed for parameter fitting, and (3) the robustness of different propagation models in indoor environments. In answering these questions, we provide guidelines for developing a practical radio mapping technique which can be used in complex indoor environments.

7.5.1 Experimental Setup

Our experiments were carried out in two indoor office buildings (see Figure 7.2; triangles represent relays and circles represent test positions) using TelosB motes. These two buildings serve as good test cases because they were constructed in different years with different materials: for example, the walls in Bryan Hall contain steel rebars that attenuate wireless signals, while the walls in Jolley Hall do not. The motes are equipped with CC2420 low-power radio chips, which provide an RSS indicator

Figure 7.2: Test buildings

reading for each decoded packet. All nodes in our experiment were set to 802.15.4 channel 26, which does not overlap with the buildings' 802.11g network.

The experimental setup is motivated by our interest in supporting robust data collection from mobile users. Accordingly, we aim to ensure that at least one relay node is capable of receiving data from a user standing in any location. A total of 28 and 45 motes were deployed close to the ceilings of Jolley and Bryan Halls, respectively, representing the locations where relays have been deployed. We used these nodes to record the link quality at when a test node is placed at numerous locations in the two buildings (104 locations in Jolley and 64 locations in Bryan). At each measurement location, a sender node broadcasted packets at the eight power levels available on the CC2420 radios. The Jolley dataset was collected with the sender placed 1.5m off the ground on a tripod, while the Bryan dataset used plastic cups 15 cm off the ground. The relays recorded the RSS reading and sequence number of each successfully decoded packet, which was relayed to a central database through a wired back channel. Each data set was collected during the night over two consecutive days.

To compensate for errors in the CC2420's raw RSS readings, we calibrated the RSS data in a similar fashion as [23], using a calibration curve provided by the authors. After calibration, the collected data is divided into training and testing sets. To account for uneven spatial distribution in our measurements, we construct the training and testing sets as follows. A number of points are generated uniformly over the 2D floor plan, and the links with senders closest to these points are selected as part of

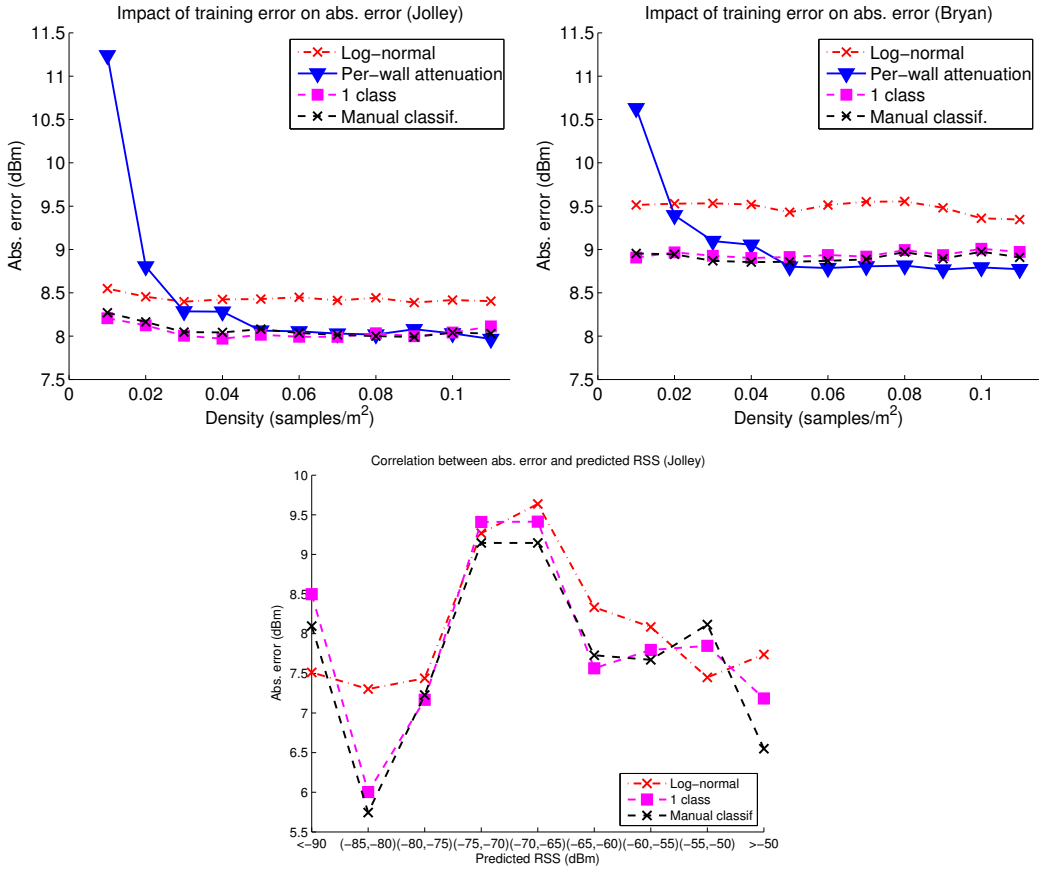


Figure 7.3: Comparison of propagation models

the training set. The testing set is generated from the remaining links in a similar fashion. We vary the size of the training set by varying the number of randomly generated points, with densitys ranging from 0.01 to 0.11 samples/m^2 in increments of 0.01.

Unless mentioned otherwise, the presented results are averages of 10 randomly generated training sets. We evaluate the performance of various models based on the 80th percentile of the absolute error between predicted and actual RSS values.

7.5.2 Effect of Walls

First, we evaluate the effectiveness of including walls in our radio propagation models by comparing the performance of models which incorporate topological information

against the log-normal model. Figure 7.3 presents the error for these models, including three approaches that include wall attenuation: treating each wall as an independent variable (*per-wall attenuation*), assuming that walls are of the same construction material (*1 wall class*), and manually labeling the type of each wall based on architectural knowledge (*manual classification*). The 1-wall class and manual classification models consistently outperform the log-normal model in both environments, with 4.7%–8.4% lower error regardless of the amount of training data available.

The overall reductions in error are modest. However, we note that not all links are equally important for overall coverage predictions. Rather, the most important links are those close to the coverage boundary, which have an RSS in the transitional region. As discussed in detail in Section 7.6, the transitional region in our testbed occurs the RSS range of $[-87, -80]$ dBm. In this region, we have found that the models which model wall attenuation can significantly outperform models which do not. This effect is illustrated in Figure 7.3(c), which plots error as a function of predicted RSS for the Jolley dataset with a density of 0.1. Within the transitional region, the models which incorporate walls outperform the log-normal model by as much as 22.2%.

We also observed that adding more training data only slightly improves most of the models' performance. The per-wall attenuation model is the exception, improving by as much as 30% when more training data is provided. This is because the per-wall attenuation model has about 100 parameters that require large amounts of data to accurately estimate, whereas the other models have few parameters that can be fit well using relatively little data. We observe that the per-wall attenuation model may outperform the other wall models when given enough training data.

Summary: *The number of parameters in a model must be tuned to match the amount of available training data.*

7.5.3 Automatic Wall Classification

The previous experiment showed the benefits of using wall information and the pitfalls of using models with numerous parameters for limited training data. In this section, we consider models which automatically classify walls into a small number of classes.

By constraining the number of classes, we hypothesize that we can improve prediction accuracy without requiring large training sets.

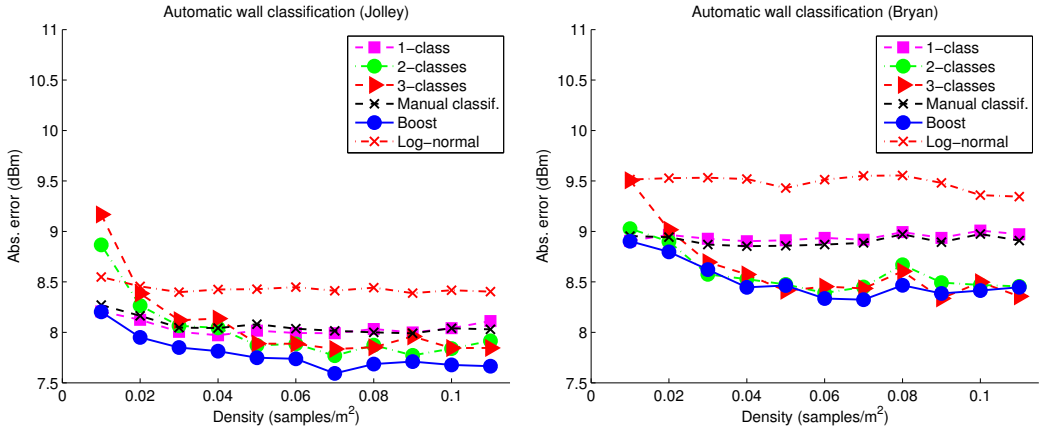


Figure 7.4: Automatic wall classification

Figure 7.4 compares the estimation accuracy of the automatic wall-classification model with 1–3 classes of walls. For comparison, we also include the per-wall model and the manual wall classification model. When little training data is available, using fewer wall classes improves the predicted accuracy for both data sets. At a density of 0.01, the 1-class model outperforms all other models, with 10.5% and 6.3% lower error in the two dataset than the 3-class model. In contrast, at a density of 0.10, the 3-class model achieves the lowest error, at 4.6% and 6.18% lower than the 1-class model. Both data sets indicate that additional wall classes are beneficial when more training data is available.

Summary: *Classifying walls into a few classes achieves the lowest error when the training data set is small; but as the amount of training data increases, more classes should be employed.*

We also note that the automatic wall classification scheme achieves lower error than the manual wall-classification scheme. In fact, on the Bryan data set, the automatic wall classification scheme has 6.2% lower error than the manual wall-classification scheme at the maximum sampling density. This is explained by the fact that the attenuation of a wall is partly determined by the additional shelving or furniture present in an office or in the room. Besides being more labor-intensive, a manual classification based purely on construction material would not capture this information.

Summary: *Automatic wall classification model achieves higher prediction accuracy without requiring the user to manually classify the walls.*

7.5.4 Boosting

Looking at the predictions from individual relays, we observed that the "best" model often depends on the location — the model with the lowest error in one room would not necessarily have the lowest error in another. Thus, we propose an approach we call *boosting*, which reduces error by combining different models' predictions. Intuitively, boosting divides the map into regions (i.e., rooms or hallways) and finds the most accurate model on a per-region basis.

Formally, boosting combines the RSS predictions and training errors of multiple models as follows. For each region R and model M, we compute the average ($\mu_{R,M}$) and standard deviation ($\sigma_{R,M}$) training error. We then select the model M_{best} which minimizes $\mu_{R,M}+2\sigma_{R,M}$ (i.e., the 95%-percentile of a normal distribution $\mathcal{N}(\mu_{R,M}, \sigma_{R,M})$). For regions which have no samples, the 1-class model is selected as a conservative choice, since it generally achieves good prediction accuracy with a small number of measurements.

As shown in Figure 7.4, the boosting procedure reduces the prediction error by as much as 8.8% over the log-normal model for the Jolley dataset, and as much as 9.7% for the Bryan dataset. More strikingly, its performance is consistently good across training sets of different sizes.

Summary: *The boosting procedure combines results from multiple models to achieve consistently good performance, independent of training data size.*

7.5.5 Impact of Sectorization

In the preceding models, we have ignored the effect of non-uniform radiation patterns. We will now explore the sectorization technique (Equation 3) that aims to improve the accuracy of the radio propagation model by modeling this effect when the automatic wall classification is used.

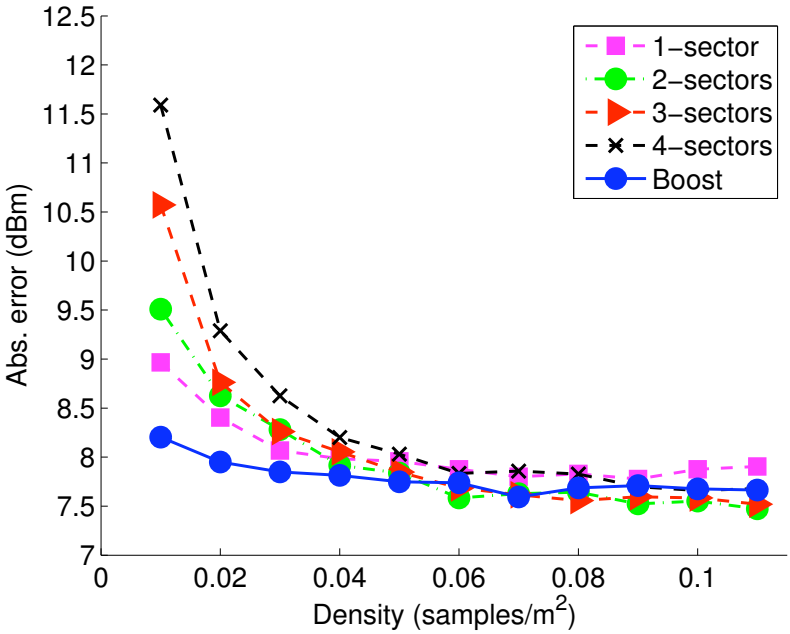


Figure 7.5: Sectorization Model

Figure 7.5 shows the prediction errors for models which consider both obstacles and directionality. In this figure, the number of wall classes is fixed at 2, and an increasing number of sectors are used. For comparison, we also include the results of the (nonsectorized) boost procedure described above.

We note that at densities lower than 0.06 samples/m^2, adding more sectors *increases* the prediction error; the sectorization models outperform the boost approach only at densities > 0.08 samples/m^2. This phenomenon is caused by the greatly increased number of parameters needed for sectorization. Rather than solving for a single parameter α, it now necessary to solve for up to 112 values for Jolley Hall and up to 256 values for Bryan Hall[12]. We emphasize that the lower densities are the most useful for radio mapping, since they represent less data that must be sampled.

Finally, we wish to explore how much decay, walls, and directionality actually contribute to wireless coverage. Figures 7.6(a) and 7.6(b) plot the impact of distance and walls for a representative wall class model. As the predicted RSS decreases, both distance and walls contribute more in absolute terms to the attenuation; this makes intuitive sense, since high-RSS links tend to be shorter and pass through fewer walls.

[12]In fact, at 5 sectors there would be more unknowns than experimental points.

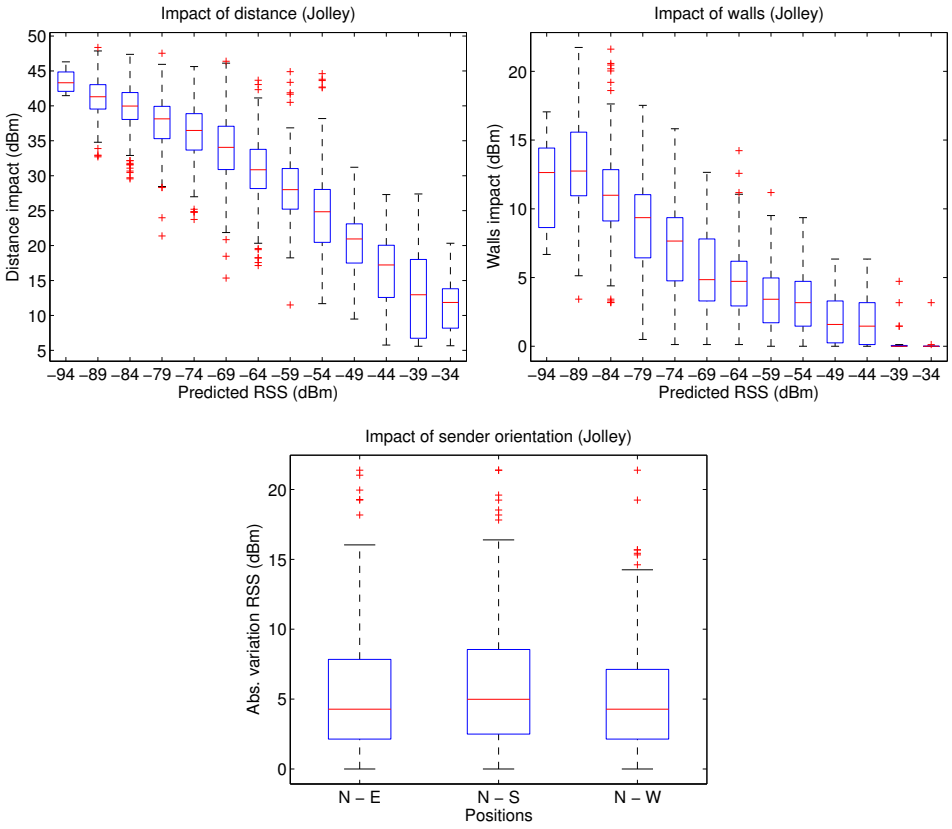


Figure 7.6: Impact of distance, wall attenuation, and antenna orientation

This effect is even more striking when considering the relative contribution. For links with high RSS, the impact of walls may be as low as 5%. For the critical links close to the coverage boundary, walls contribute to up to 25% of the overall signal loss.

To evaluate the impact of directionality on RSS, we oriented the sender in each of the four compass directions and sent a number of packets to a fixed receiver. Figure 7.6(c) presents the difference in RSS relative to when the sender is pointed North. We observe differences in the 25th, 50th, and 75th percentiles of about 2.5 dBm, 5 dBm, and 8.5 dBm, which are consistent with those observed in [145]. For links in the critical transitional region, the impact of attenuation through obstacles has twice the impact of directionality. Thus, we choose models which ignore the impact of directionality; it is more effective to use the limited training data to better fit the wall classification parameters, which are fewer and have a greater impact on signal strength near the coverage boundary.

Summary: *Antenna orientation has a smaller impact than wall attenuation for links on the boundaries of coverage regions. Moreover, sectorization techniques are suitable only when large training data sets are available.*

7.6 Radio Mapping Tool

In this section, we present our Radio Mapping Tool (RMT) for assessing network coverage. RMT is particularly beneficial for applications which require a network to cover an entire physical area. The main use case of RMT is to evaluate the coverage of an already deployed network; this can be done by computing the union of the regions covered individually by each relay. RMT may also be used to assist during the initial network deployment: an overly dense network of relays is temporarily deployed to measure network coverage, and only those necessary to cover the area are permanently installed.

RMT has several salient features. (1) In contrast to ray-tracing techniques, RMT does not require the user to specify the attenuation coefficients or construction materials of walls. Wall locations may be extracted from readily available floor plans. (2) Based on our insights from the previous sections, RMT uses the wall classification models,

which have been shown to provide accurate predictions even when few measurements are used for training. RMT combines these results through the Boost procedure. (3) RMT uses the computationally efficient algorithm presented in Section 7.4 to classify each wall into a small number of classes and determine the model parameters.

As input from the user, RMT requires the physical locations of the relays whose coverage is being predicted, the locations of walls, a set of training data, and a PRR threshold (PRR_t) that determines "good" and "bad" links. RMT includes a TinyOS application to be deployed on a single "beacon" mote, which broadcasts beacon packets at each of the 8 power levels supported by the CC2420 at places where the user wishes to collect link quality data. By design, the user does not need to collect data exhaustively; our case study in Section 7.7 used as few as 0.01 samples/m^2. A corresponding TinyOS application on the relay nodes records the RSS and sequence number of all successfully decoded packets, which RMT uses to compute the PRR and average RSS of each link.

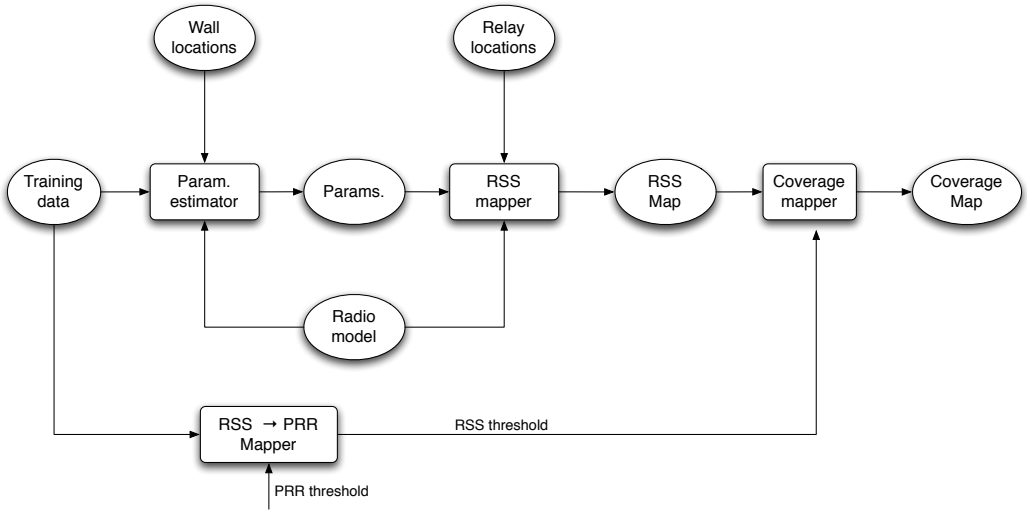


Figure 7.7: Radio Mapping Tool

RMT has four main components, as shown in Figure 7.7: a *Parameter Estimator*, an *RSS Mapper*, an *RSS-to-PRR Mapper*, and a *Coverage Mapper*. The *Parameter Estimator* uses the computationally efficient algorithm described in Section 7.4 to estimate the model's parameters. RMT fits models with 1–4 different wall types and merges their prediction using the Boosting procedure previously discussed. Based on the determined parameters, the *RSS Mapper* constructs a map of signal strength

predictions on a dense 2D grid overlaid on the floor plan. A map is individually computed for each relay using the automatic wall classification model.

The *RSS-to-PRR Mapper* determines an RSS threshold which accurately separates the "good links" (with PRR $\geq PRR_t$) from the "bad links" (with PRR $< PRR_t$). To do so, we leverage the correlation between RSS and PRR previously observed in [121]. Any RSS threshold will necessarily have some false negative rate (i.e., links incorrectly predicted as poor-quality) and false positive rate (i.e., links incorrectly predicted as good-quality). An example of this phenomenon is shown in Fig. 7.8, where the PRR threshold has been set to 80% and the false positive and false negative rates have been calculated for each possible RSS threshold. In this example, an RSS threshold of -85 dBm offers the best tradeoff: the false positive rate is 9% and false negative rate is 15%. RMT allows the user to specify the maximum acceptable false negative and false positive rates; the RSS-to-PRR mapper then automatically locates the minimum RSS which satisfies both of these criteria in the training data, or reports an error when no such RSS threshold exists. We note that from the perspective of coverage prediction, it is desirable to have a low false positive rate, and it should arguably be configured conservatively. Nevertheless, an overly conservative threshold could result in overdeployment, which increases the monetary cost of the deployment and may increase channel contention. We note that the *RSS-to-PRR Mapper* is designed under the assumption that the noise levels observed during the measurements are representative of normal network operations. This could be improved with a better model of background traffic, such as the one proposed in [80].

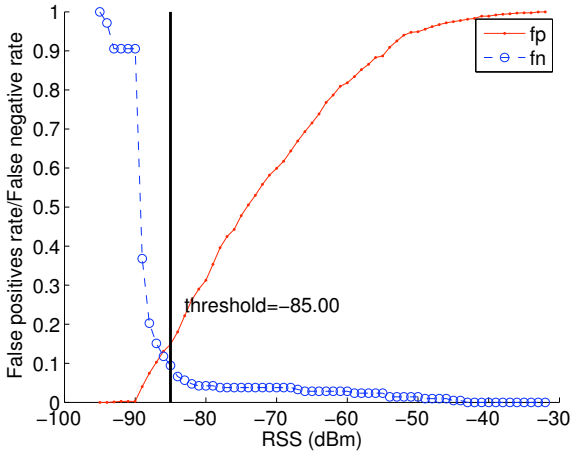


Figure 7.8: Selecting the RSS threshold

Finally, the *Coverage Mapper* uses this threshold to convert the RSS map into a binary coverage decision at each grid point; Figure 7.9 shows a sample output. By precomputing as much data as possible (e.g., the walls between each relay and grid location), RMT can train the models and make predictions within minutes.

7.7 Empirical Evaluation of RMT

In this section, we analyze the performance of the RMT on the data sets previously collected for our empirical study. We begin by assessing RMT's performance through a case study which highlights RMT's accuracy and the intuitive nature of the outputted radio maps.

We characterize the accuracy of RMT's coverage predictions by its resulting false positive and false negative rates. In contrast to the previous section, the false positive and false negative rates discussed here refer to the prediction coverage rather than the RSS threshold. In this context, a false postive occurs when RMT predicts coverage where there is none; similarly, a false negative occurs when RMT predicts no coverage but ground truth data indicates otherwise.

7.7.1 Representative Example

The case-study is designed to emulate the use of RMT to predict the coverage of one relay in Jolley Hall. In order to illustrate the efficacy of our automatic wall-classification model, we present results with the normal RMT (which uses the automatic wall classification model with Boost) as well as with a version of RMT that has been modified to use the basic log-normal model. To highlight RMT's accuracy when using only a small amount of training data, we choose a sampling density of 0.02 samples/m^2. The data is divided into training and testing sets through the same sampling strategy described in Section 7.5. For the purposes of this study, we define a "good link" to have a PRR higher than 80%. Using the RSS threshold selection technique previously discussed, a RSS threshold of -85 dBm was selected.

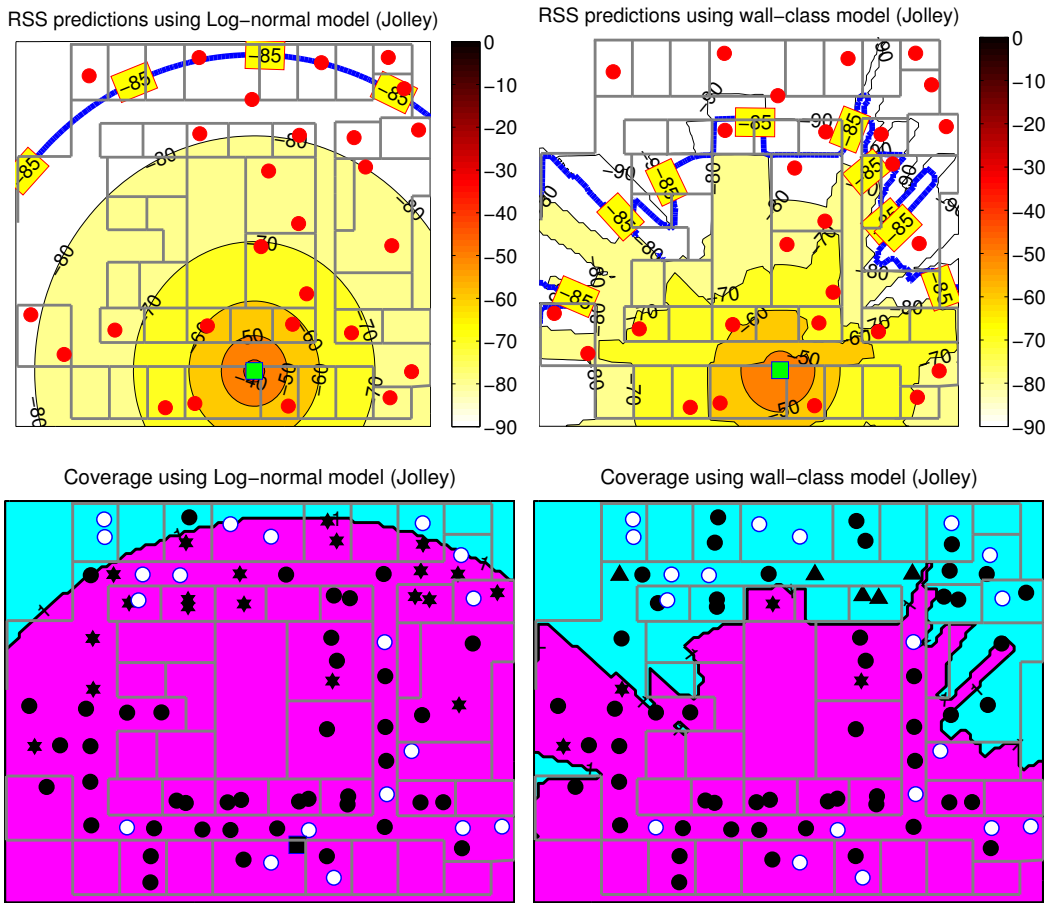


Figure 7.9: Example predictions using Radio Mapping Tool

Figure 7.9(a) plots the RSS predictions when the log-normal model is used. Since the log-normal model does not account for wall attenuation, the contour graph consists of concentric circles. RMT also plotted the −85 dBm line that delineates the relay's coverage area. It is worth highlighting the small number of samples which was used for training, shown as red dots. Moreover, we have only a few measurements in each room; other radio mapping techniques require every wall to be independently measured. Figure 7.9(c) shows the RSS predictions made by RMT using Boost. The predictions clearly indicate the strong impact of walls; the finger-like projections are caused by signals passing through different numbers of walls.

Figure 7.9(b) shows the coverage map for the log-normal model. Here, the white circles represent training data, the black circles indicate correct predictions, and the

stars and triangles denote false positives and false negatives, respectively. Figure 7.9(d) plots the corresponding coverage map predicted when using Boost.

The log-normal model has 20 false positives, which are particularly disconcerting since they indicate coverage in regions where there is actually none. In contrast, RMT reduces the number of false positives from 20 to 4. This is particularly clear toward the top of the predicted coverage area, where the coverage area stops at the intersection with a wall. We note that most of these 4 false positive locations are close to the predicted coverage border. We expect that the coverage prediction could be further improved by targeted sampling near the border. This highlights the use of RMT as an interactive tool to guide the user about where to collect additional coverage measurements.

Due to the log-normal model being overly optimistic, it predicts 1 false negative compared to 5 for the wall-based model. We note that some false negatives should be expected, since a threshold of -85 dBm leads to a 15% false negative rate when mapping RSS to PRR.

7.7.2 Detailed Empirical Results

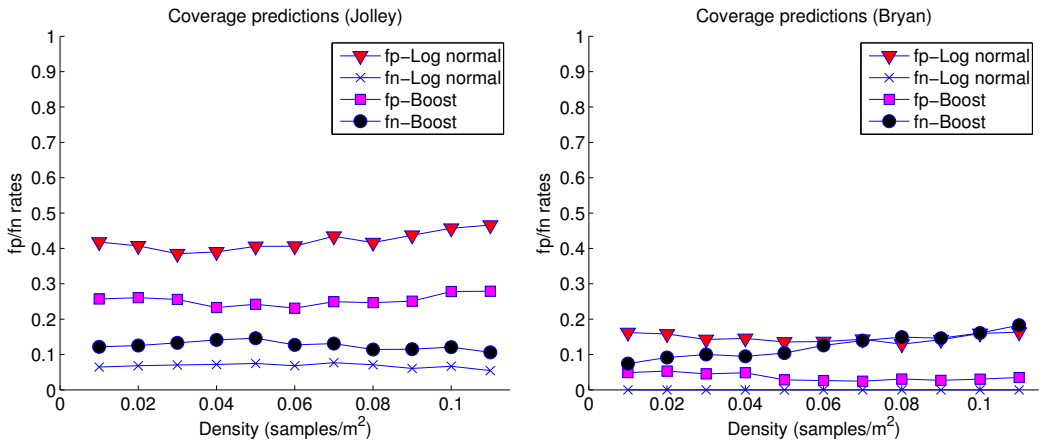


Figure 7.10: Coverage prediction accuracy

Next, we statistically analyze RMT's overall performance on both buildings' data sets. For this analysis, we fix the PRR threshold to 80% and set the RSS false positive rate threshold to 10% for both data sets. This resulted in an RSS threshold of -85 dBm

and -87 dBm for Jolley and Bryan, respectively. Additionally, we varied the sampling density and observed the impact on RMT's performance.

Figure 7.10 plots the false negative and false positive rates for RMT when using the log-normal and Boost-based models. First, consider the results from the Jolley dataset. As seen in the case study, the log-normal model suffers from numerous false postives, with a false positive rate of 38%–46%. In contrast, RMT using the Boost procedure has a false positive rate between 23%–27%, representing a reduction in the false positive rate by up to 39% at a density of 0.01.

Again, the log-normal model achieves the lowest false positive rate (6.8%) by incurring a high false negative rate. The false negative rate for RMT was 10%–12%, which is comparable with the 10% false positive rate imposed on the RSS threshold. As previously noted, a moderate increase in false negatives may be acceptable, since it would result in a slightly denser network.

The results from the Bryan data set paint a similar picture. Using the Boost procedure gives RMT a significantly lower false positive rate than the log-normal model. When the sampling rate is 0.01 samples/m^2, the Boost-based method reduced the false positive rate by as much as 54% (from 16.21% down to 7.42%). At the same sampling density, the Boost-based method has false negative rate of 16%, compared to a false negative rate of 0% for the log-normal model. Again, the false negative of zero occurs because the log-normal model significantly overestimates coverage area.

We conclude that our Boost-based approach may reduce false positive rates by as much as 54%, and achieved a false negative rate comparable to the user-specified constraints on the RSS threshold. Moreover, despite the two different radio propagation characteristics, RMT with Boost achieved consistently good performance across two different buildings.

7.8 Summary

Radio mapping is a challenging problem for real indoor environments due to signal attenuation through walls, complex signal propagation behavior, and the need to

reduce the number of sampling measurements. This chapter addresses this important challenge by developing a practical and effective radio mapping approach for indoor environments.

We first perform an in-depth empirical analysis of several signal propagation models in an office building. Our analysis shows the importance of balancing the accuracy of the model against the number of model parameters that need be estimated based on limited measurement. Our empirical results identify the wall-classification model family as the most practical and effective for indoor environments.

We then propose a practical algorithm to predict the RSS between different locations based on a small number of measurements. A key novelty of our algorithm lies in its ability to automatically classify walls into a small number of classes with different degrees of signal attenuation, and to automatically select the best number of wall classes on a per-region basis. Empirical results show that our automatic wall classification scheme results in more accurate RSS prediction than a manual classification based on architectural knowledge.

We have developed a practical Radio Mapping Tool to predict the radio coverage of relay placements. RMT has several salient features. (1) It requires minimal information about the indoor environment. The only knowledge about the environment that RMT needs are the wall locations, which may be extracted from existing floor plans. (2) RMT can accurately predict radio coverage based on a small number of measurements, which can significantly reduce the cost of network deployment and maintenance. (3) RMT features computationally efficient algorithms that allow users to quickly assess and adjust the coverage of a potential relay placement.

An empirical evaluation in two office buildings shows that RMT achieves as much as 54% fewer false positives compared to the log-normal model based on a sampling density of only 0.01 samples/m^2. Our results demonstrate that RMT is a practical tool which can be used to facilitate the efficient deployment and robust operation of wireless sensor networks for indoor environments.

Chapter 8

Conclusions and Future Work

This thesis has focused on the development of real-time infrastructure for data collection applications such a structural health monitoring and patient monitoring. The first part of the thesis considers the problem of supporting real-time communication in systems that employ contention-based media access. The protocols proposed in Chapters 2 and 3 describe novel approaches to achieving both energy efficiency and supporting real-time communication. The development of real-time communication support for applications employing time-division media access is considered in Chapters 4 and 5. Novel transmission scheduling and static prioritization schemes for real-time data collection are proposed. By taking advantage of the predictable performance provided by transmission scheduling approaches, we are able to characterize the performance of a system in terms of maximum supported throughput, maximum latency, and energy consumption. The proposed techniques are designed to handle dynamic application workloads as well as dynamic network topologies. This work promises to be a key building block for the next generation of cyber-physical systems that require predictable performance over large-scale wireless sensor networks.

The remainder of the thesis focused on the design, implementation, and validation of a patient monitoring system. The system features a clean design based on the unique characteristics of patient monitoring in general hospital units which include the need to develop a low-cost reliable system and to support patient mobility. We developed a simple yet highly reliable solution for handling patient mobility. The key contribution of this work is the empirical validation of the system through a clinical trial performed in a step-down cardiology unit at Barnes-Jewish Hospital. The clinical study is the first long-term and large-scale deployment which provides

evidence regarding the feasibility of using low-power and low-cost 802.15.4 wireless technology in clinical environments.

8.1 Future Work

I believe that the work developed as part of this thesis can be the basis of significant future research. My work on real-time communication for sensor network applications could be extended in several ways:

- Even though data collection is the most prevalent communication pattern in sensor networks, there are other applications that would benefit from real-time communication but cannot express their communication interests as queries. A more diverse set of communication patterns (e.g., flows, dissemination) should be considered. A more challenging aspect would be to support a mix communication patterns.
- Currently, the developed algorithms have been validated through simulation. As future work, I propose to deploy the proposed protocols as part of a real system. I expect this to pose particularly interesting challenges in how to best architect the communication stack to enable real-time communication.
- The framework developed for real-time communication enables a system developer to determine the performance of a system at deployment time. Due to the predictable performance of the system, it would be easy to evaluate the impact of changes in various system parameters (e.g., data rates, priorities, etc). This would be a solid foundation for building support for adapting application behavior as to meet various system requirements including life time or reliability.

The patient monitoring system could be improved in the following ways:

- The reliability of the system gives us confidence that we may detect clinical deterioration. As a next step, it would be particularly exciting to close the loop by implementing clinical deterioration algorithms on the base station.

- Understanding the correlation between the sampling rate and the accuracy of predicting clinical deterioration is an another research topic. Note that this may have profound implications on how power management is performed on patient nodes. Moreover, we are also interested in incorporating more sensors such as breathing rate to provide a richer set of measurements for clinical deterioration.

- Even though the network reliability was high in our system, we failures occurred it was particularly difficult to track their source. As future work I propose to develop a diagnosis plane that has standardized interfaces for characterizing common failures from each component of the communication stack including media access, link layer, routing, and power management. The information provided by the diagnosis plane could be interpreted by expert systems to determine higher-level failures such as network congestion or network partition. To accomplish this, information from different layers in the protocol stack and, in some cases, from multiple nodes may be required.

References

[1] Crossbow, inc., mica2 mote. `http://www.xbow.com/Products/productsdetails.aspx?sid=72`, April 19 2006.

[2] I. Aad and C. Castelluccia. Differentiation mechanisms for ieee 802.11. In *Proceedings of INFOCOM*, volume 1, pages 209–218 vol.1, 2001.

[3] T.R. Abdelzaher, S. Prabh, and R. Kiran. On real-time capacity limits of multihop wireless sensor networks. In *Proceedings of RTSS*, pages 359–370, Dec. 2004.

[4] Gahng-Seop Ahn, A.T. Campbell, A. Veres, and Li-Hsiang Sun. SWAN: service differentiation in stateless wireless ad hoc networks. In *Proceedings of INFOCOM*, volume 2, pages 457 – 466 vol.2, 2002.

[5] Gahng-Seop Ahn, Se Gi Hong, Emiliano Miluzzo, Andrew T. Campbell, and Francesca Cuomo. Funneling-MAC: a localized, sink-oriented mac for boosting fidelity in sensor networks. In *Proceedings of SenSys*, pages 293–306, 2006.

[6] J.B. Andersen, T.S. Rappaport, and S. Yoshida. Propagation measurements and models for wireless communications channels. *IEEE Communications Magazine*, 33(1):42–49, Jan 1995.

[7] E. Arikan. Some complexity results about packet radio networks. *NASA STI/Recon Technical Report N*, 83, March 1983.

[8] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, Sep 1993.

[9] Lichun Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for ad hoc networks. In *Proceedings of MobiCom*, pages 210–221, 2001.

[10] M. Barry, A.T. Campbell, and A. Veres. Distributed control algorithms for service differentiation in wireless packet networks. In *Proceedings of INFOCOM*, volume 1, pages 582–590 vol.1, 2001.

[11] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):299–316, Jun 2000.

[12] R. Bernhardt. Macroscopic diversity in frequency reuse radio systems. *IEEE Journal on Selected Areas in Communications*, 5(5):862–870, Jun 1987.

[13] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[14] T A Brennan, L L Leape, N M Laird, L Hebert, A R Localio, A G Lawthers, J P Newhouse, P C Weiler, and H H Hiatt. Incidence of adverse events and negligence in hospitalized patients. results of the harvard medical practice study i. *N Engl J Med*, 324(6):370–6, Feb 1991.

[15] B.D. Bui, R. Pellizzoni, M. Caccamo, C.F. Cheah, and A. Tzakis. Soft real-time chains for multi-hop wireless ad-hoc networks. In *Proceedings of RTAS*, pages 69–80, April 2007.

[16] M D Buist, E Jarmolowski, P R Burton, S A Bernard, B P Waxman, and J Anderson. Recognising clinical instability in hospital patients before cardiac arrest or unplanned admission to intensive care. a pilot study in a tertiary-care hospital. *Med J Aust*, 171(1):22–5, Jul 1999.

[17] M. Caccamo, L.Y. Zhang, Lui Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proceedings of RTSS*, pages 39–48, 2002.

[18] A. Cerpa, J.L. Wong, L. Kuang, Potkonjak, and D. M., Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of IPSN*, pages 81 – 88, April 2005.

[19] Jae-Hwan Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of INFOCOM*, volume 1, pages 22–31 vol.1, 2000.

[20] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of MobiCom*, pages 85–96, 2001.

[21] Bor-rong Chen, Kiran-Kumar Muniswamy-Reddy, and Matt Welsh. Ad-hoc multicast routing on resource-limited sensor nodes. In *Proceedings of REALMAN*, pages 87–94, 2006.

[22] Shigang Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1488–1505, Aug 1999.

[23] Yin Chen and Andreas Terzis. Calibrating RSSI measurements for 802.15.4 radios. Technical report, John Hopkins University, 2009.

[24] Krishna Chintalapudi, Jeongyeup Paek, Omprakash Gnawali, Tat S. Fu, Karthik Dantu, John Caffrey, Ramesh Govindan, Erik Johnson, and Sami Masri. Structural damage detection and localization using NETSHM. In *Proceedings of IPSN*, pages 475–482, 2006.

[25] O. Chipara, Z. He, Guoling Xing, Qin Chen, Xiaorui Wang, Chenyang Lu, J. Stankovic, and T. Abdelzaher. Real-time power-aware routing in sensor networks. In *Proceedings of IWQoS*, pages 83–92, June 2006.

[26] Octav Chipara, Chris Brooks, Sangeeta Bhattacharya, Chenyang Lu, Roger Chamberlain, Gruia-Catalin Roman, and Thomas C. Bailey. Reliable real-time clinical monitoring using sensor network technology. In *Proceedings of AMIA*, 2009.

[27] Octav Chipara, Chenyang Lu, and Gruia-Catalin Roman. Efficient power management based on application timing semantics for wireless sensor networks. In *Proceedings of ICDCS*, pages 361–370, 2005.

[28] Octav Chipara, Chenyang Lu, and John Stankovic. Dynamic conflict-free query scheduling for wireless sensor networks. In *Proceedings of ICNP*, pages 321–331, 2006.

[29] Imrich Chlamtac and András Faragó. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions Networking*, 2(1):23–29, 1994.

[30] Y. Chu and A. Ganz. A mobile teletrauma system using 3G networks. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):456–462, Dec. 2004.

[31] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, 38(10):1353–1361, Oct 1989.

[32] The Joint Commission. 2008 national patient safety goals.

[33] Dorothy W. Curtis, Esteban J. Pino, Jacob M. Bailey, Eugene I. Shih, Jason Waterman, Staal A. Vinterbo, Thomas O. Stair, John V. Guttag, Robert A. Greenes, and Lucila Ohno-Machado. Smart–an integrated wireless system for monitoring unattended patients. *Journal of the American Medical Informatics Association*, 15(1):44 – 53, 2008.

[34] Erik D. Demaine, Alejandro Lopez-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of ESA*, pages 348–360, 2002.

[35] Sheetalkumar Doshi, Shweta Bhandare, and Timothy X Brown. An on-demand minimum energy routing protocol for a wireless ad hoc network. *Mobile Computing Communication Review*, 6(3):50–66, 2002.

[36] M. D'Souza, T. Wark, and M. Ros. Wireless localisation network for patient tracking. *Proceedings of ISSNIP*, pages 79–84, 2008.

[37] A. Ephremides and T.V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 38(4), 1990.

[38] T. Facchinetti, L. Almeida, G. C. Buttazzo, and C. Marchini. Real-time resource reservation protocol for wireless mobile ad hoc networks. *Proceedings of RTSS*, pages 382–391, 2004.

[39] Qing Fang, Jie Gao, and L. J. Guibas. Locating and bypassing routing holes in sensor networks. In *Proceedings of INFOCOM*, volume 4, pages 2458–2468 vol.4, 2004.

[40] E. Felemban, C. G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. In *Proceedings of INFOCOM*, volume 4, pages 2646–2657 vol. 4, 2005.

[41] Rodrigo Fonseca, Omprakash Gnawali, and Kyle Jamiesonand Philip Levis. Four bit wireless link estimation. In *Proceedings of HotNets*, 2007.

[42] Steven J. Fortune, David M. Gay, Brian W. Kernighan, Orlando Landron, Reinaldo A. Valenzuela, and Margaret H. Wright. Wise design of indoor wireless systems: Practical computation and optimization. *IEEE Computational Science and Engineering*, 1995.

[43] Mohamed R. Fouad, Sonia Fahmy, and Gopal Pandurangan. Latency-sensitive power control for wireless ad-hoc networks. In *Proceedings of Q2SWinet*, pages 31–38, 2005.

[44] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, UCLA Computer Science Department, 2002.

[45] Tia Gao, T Massey, L Selavo, D Crawford, Bor rong Chen, K Lorincz, V Shnayder, L Hauenstein, F Dabiri, J Jeng, A Chanmugam, D White, M Sarrafzadeh, and M Welsh. The advanced health and disaster aid network: A light-weight wireless medical system for triage. *IEEE Transactions on Biomedical Circuits and Systems*, Aug 2007.

[46] Ye Ge and J. Hou. An analytical model for service differentiation in ieee 802.11. In *Proceedings of ICC*, volume 2, pages 1157–1162 vol.2, 2003.

[47] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of SenSys*, pages 1–14, New York, NY, USA, 2009. ACM.

[48] J. Gomez and A. T. Campbell. A case for variable-range transmission power control in wireless multihop networks. In *Proceedings of INFOCOM*, volume 2, pages 1425–1436 vol.2, 2004.

[49] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[50] H. Hashemi. The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968, 1993.

[51] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *Proceedings of ICDCS*, page 46, 2003.

[52] Tian He, Pascal Vicaire, Ting Yan, Liqian Luo, Lin Gu, Gang Zhou, Radu Stoleru, Qing Cao, John A. Stankovic, and Tarek Abdelzaher. Achieving real-time target tracking usingwireless sensor networks. In *Proceedings of RTAS*, pages 37–48, 2006.

[53] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001.

[54] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[55] Barbara Hohlt, Lance Doherty, and Eric Brewer. Flexible power scheduling for sensor networks. In *Proceedings of IPSN*, pages 205–214, 2004.

[56] Marilyn Hravnak, Leslie Edwards, Amy Clontz, Cynthia Valenta, Michael A Devita, and Michael R Pinsky. Defining the incidence of cardiorespiratory instability in patients in step-down units using an electronic integrated monitoring system. *Archives of Internal Medicine*, 168(12):1300–8, Jun 2008.

[57] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of SenSys*, pages 134–147, 2004.

[58] Joengmin Hwang, Tian He, and Yongdae Kim. Exploring in-situ sensing irregularity in wireless sensor networks. In *Proceedings of SenSys*, pages 289–303, 2007.

[59] IEEE. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[60] IEEE 802.11 WG. Wireless MAC and physical specifications: MAC enhancements for QoS, February 2003.

[61] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions Networking*, 11(1):2–16, 2003.

[62] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM*, pages 314–329, 1988.

[63] Ji-Her Ju and Victor O. K. Li. An optimal topology-transparent scheduling method in multihop packet radio networks. *IEEE/ACM Transactions Networking*, 6(3):298–306, 1998.

[64] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of MobiCom*, pages 200–209, 2001.

[65] Kyriakos Karenos, Vana Kalogeraki, and Srikanth V. Krishnamurthy. A rate control framework for supporting multiple classes of traffic in sensor networks. In *Proceedings of RTSS*, pages 287–297, 2005.

[66] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of MobiCom*, pages 243–254, 2000.

[67] Abel Kho, David Rotz, Kinan Alrahi, Wendy Cárdenas, Kristin Ramsey, David Liebovitz, Gary Noskin, and Chuck Watts. Utility of commonly captured data from an EHR to identify hospitalized patients at risk for clinical deterioration. *Proceedings of AMIA*, pages 404–8, Dec 2007.

[68] James P Killeen, Theodore C Chan, Colleen Buono, William G Griswold, and Leslie A Lenert. A wireless first responder handheld device for rapid triage, patient assessment and documentation during mass casualty incidents. In *Proceedings of AMIA*, pages 429–433, 2006.

[69] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of IPSN*, pages 254–263, 2007.

[70] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *Proceedings of NSDI*, pages 217–230, 2005.

[71] Naoto Kimura and Shahram Latifi. A survey on data compression in wireless sensor networks. In *Proceedings of ITCC*, pages 8–13, 2005.

[72] JeongGil Ko, Răzvan Musăloiu-Elefteri, Jong Hyun Lim, Yin Chen, Andreas Terzis, Tia Gao, Walt Destler, and Leo Selavo. MEDiSN: medical emergency detection in sensor networks. In *Proceedings of SenSys*, pages 361–362, 2008.

[73] A. Koubaa, M. Alves, and E. Tovar. i-game: an implicit gts allocation mechanism in ieee 802.15.4 for time-sensitive wireless sensor networks. In *Proceedings of ECRTS*, pages 10 pp.–192, 2006.

[74] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *Proceedings of IPSN*, pages 2–10, 2006.

[75] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proceedings of SenSys*, pages 64–75, 2005.

[76] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proceedings of PODC*, pages 63–72, 2003.

[77] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Algorithmic aspects of capacity in wireless networks. In *Proceedings of SIGMETRICS*, pages 133–144, 2005.

[78] Mathieu Lacage. NS2 802.11b/e support. http://yans.inria.fr/ns-2-80211/.

[79] L L Leape, T A Brennan, N Laird, A G Lawthers, A R Localio, B A Barnes, L Hebert, J P Newhouse, P C Weiler, and H Hiatt. The nature of adverse events in hospitalized patients. results of the harvard medical practice study ii. *N Engl J Med*, 324(6):377–84, Feb 1991.

[80] HyungJune Lee, Alberto Cerpa, and Philip Levis. Improving wireless simulation through noise modeling. In *Proceedings of IPSN*, pages 21–30, 2007.

[81] Huan Li, Prashant Shenoy, and Krithi Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *Proceedings of RTAS*, pages 415–425, 2005.

[82] Qun Li, Javed Aslam, and Daniela Rus. Online power-aware routing in wireless ad-hoc networks. In *Proceedings of MobiCom*, pages 97–107, 2001.

[83] Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, John A. Stankovic, and Tian He. ATPC: adaptive transmission power control for wireless sensor networks. In *Proceedings of SenSys*, pages 223–236, 2006.

[84] Yuan-Hsiang Lin, I-Chien Jan, P.C.-I. Ko, Yen-Yu Chen, Jau-Min Wong, and Gwo-Jen Jan. A wireless PDA-based physiological monitoring system for patient transport. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):439–447, 2004.

[85] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and ZebraNet. In *Proceedings of MobiSys*, pages 256–269, 2004.

[86] Konrad Lorincz, David J Malan, Thaddeus R F Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, and Matt Welsh. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, Sep 2004.

[87] Chenyang Lu, Guoliang Xing, Octav Chipara, Chien-Liang Fok, and Sangeeta Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In *Proceedings of ICDCS*, pages 381–390, 2005.

[88] Jerome P. Lynch and Kenneth J. Loh. A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring. *The Shock and Vibration Digest*, 38(2):91–128, 2006.

[89] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *Proceedings of OSDI*, 36(SI):131–146, 2002.

[90] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions Database Systems*, 30(1):122–173, 2005.

[91] R. Maheshwari, Jing Cao, and S. R. Das. Physical interference modeling for transmission scheduling on commodity wifi hardware. In *Proceedings of INFOCOM*, pages 2661–2665, 2009.

[92] Ritesh Maheshwari, Shweta Jain, and Samir R. Das. A measurement study of interference modeling and scheduling in low-power wireless networks. In *Proceedings of SenSys*, pages 141–154, 2008.

[93] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of WSNA*, pages 88–97, 2002.

[94] Rahul Mangharam, Anthony Rowe, Raj Rajkumar, and Ryohei Suzuki. Voice over sensor networks. In *Proceedings of RTSS*, pages 291–302, 2006.

[95] S. Mangold, Sunghyun Choi, G. R. Hiertz, O. Klein, and B. Walke. Analysis of IEEE 802.11e for QoS support in wireless LANs. *IEEE Wireless Communications*, 10(6):40–50, 2003.

[96] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of SenSys*, pages 39–49, 2004.

[97] J. W. McKown and Jr. Hamilton, R. L. Ray tracing as a design tool for radio networks. *IEEE Network*, 5(6):27–30, 1991.

[98] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee. Distributed mechanisms for quality of service in wireless lans. *IEEE Wireless Communications*, 10(3):26–34, 2003.

[99] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of SenSys*, pages 95–107, 2004.

[100] Joseph Polastre, Jonathan Hui, Philip Levis, Jerry Zhao, David Culler, Scott Shenker, and Ion Stoica. A unifying link abstraction for wireless sensor networks. In *Proceedings of SenSys*, pages 76–89, 2005.

[101] Joseph Polastre, Robert Szewczyk, Cory Sharp, and David Culler. The mote revolution: Low power wireless sensor network devices. In *Proceedings of Hot Chips*, 2004.

[102] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of SenSys*, pages 181–192, 2003.

[103] R. Ramaswami and K. K. Parhi. Distributed scheduling of broadcasts in a radio network. In *Proceedings of INFOCOM*, pages 497–504 vol.2, 1989.

[104] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *Proceedings of MASS*, pages 224–234, 2004.

[105] I. Rhee, A. Warrier, Jeongki Min, and Lisong Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 8(10):1384–1396, 2009.

[106] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-mac: a hybrid mac for wireless sensor networks. In *Proceedings of SenSys*, pages 90–101, 2005.

[107] Joshua Robinson, Ram Swaminathan, and Edward W. Knightly. Assessment of urban-scale wireless networks with a small number of measurements. In *Proceedings of MobiCom*, pages 187–198, 2008.

[108] Paul Rubel, Jocelyne Fayn, Giandomenico Nollo, Deodato Assanelli, Bo Li, Lioara Restier, Stefano Adami, Sebastien Arod, Hussein Atoui, Mattias Ohlsson, Lucas Simon-Chautemps, David Telisson, Cesare Malossi, Gian-Luca Ziliani, Alfredo Galassi, Lars Edenbrandt, and Philippe Chevalier. Toward personal ehealth in cardiology. results from the epi-medics telemedicine project. *Journal of Electrocardioly*, 38(4):100–106, Oct 2005.

[109] A. Sankar and Zhen Liu. Maximum lifetime routing in wireless ad-hoc networks. In *Proceedings of INFOCOM*, volume 2, pages 1089–1097 vol.2, 2004.

[110] K.R. Schaubach, N.J. Davis, and T.S. Rappaport. A ray tracing method for predicting path loss and delay spread in microcellular environments. In *Proceedings of VTC*, volume 2, pages 932 – 935, 1992.

[111] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Proceedings of SenSys*, pages 108–121, 2004.

[112] S.Y. Seidel, T.S. Rappaport, M.J. Feuerstein, K.L. Blackard, and L. Grindstaff. The impact of surrounding buildings on propagation for wireless in-building personal communications system design. In *IEEE Vehicular Technology Conference*, 1992.

[113] Mo Sha, Guoliang Xing, Gang Zhou, Shucheng Liu, and Xiaorui Wang. C-MAC: Model-driven concurrent medium access control for wireless sensor networks. In *Proceedings of INFOCOM*, pages 1845–1853, 2009.

[114] Eugene I. Shih, Ali H. Shoeb, and John V. Guttag. Sensor selection for energy-efficient ambulatory medical monitoring. In *Proceedings of MobiSys*, pages 347–358, 2009.

[115] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *Proceedings of IEEE Aerospace Conference*, 2003.

[116] J E Sinex. Pulse oximetry: principles and limitations. *American Journal of Emergency Medicine*, 17(1):59–67, Jan 1999.

[117] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of MobiCom*, pages 181–190, 1998.

[118] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. In *Proceedings of INFOCOM*, volume 1, pages 202–209 vol.1, 1999.

[119] Dongjin Son, B. Krishnamachari, and J. Heidemann. Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks. *Proceedings of SECON*, pages 289–298, 2004.

[120] Jianping Song, Song Han, A. K. Mok, Deji Chen, M. Lucas, and M. Nixon. WirelessHART: Applying wireless technology in real-time industrial process control. In *Proceedings of RTAS*, pages 377–386, 2008.

[121] Kannan Srinivasan and Philip Levis. RSSI is under appreciated. In *Proceedings of EmNets*, 2006.

[122] Tsenka Stoyanova, Fotis Kerasiotis, Aggeliki Prayati, and George Papadopoulos. Evaluation of impact factors on RSS accuracy for localization and tracking applications. In *Proceedings of MobiWac*, pages 9–16, 2007.

[123] Jun Sun. *Fixed-Priority End-To-End Scheduling in Distributed Real-Time Systems*. PhD thesis, UIUC, 1997.

[124] Yu-Chee Tseng, Chih-Shun Hsu, and Ten-Yueng Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *Proceedings of INFOCOM*, volume 1, pages 200–209 vol.1, 2002.

[125] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of SenSys*, pages 171–180, 2003.

[126] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *Proceedings of SenSys*, pages 266–279, 2003.

[127] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of SenSys*, pages 28–39, 2003.

[128] R M Wilson, W B Runciman, R W Gibberd, B T Harrison, L Newby, and J D Hamilton. The quality in australian health care study. *Med J Aust*, 163(9):458–71, Nov 1995.

[129] Alec Woo and David E. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of MobiCom*, pages 221–235, 2001.

[130] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of SenSys*, pages 14–27, 2003.

[131] Alec Woo, Kamin Whitehouse, Fred Jiang, Joseph Polastre, and David Culler. The shadowing phenomenon: implications of receiving during a collision. In *UCB Technical Report*, 2004.

[132] A. Wood, J. Stankovic, G. Virone, L. Selavo, Zhimin He, Qiuhua Cao, Thao Doan, Yafeng Wu, Lei Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 2008.

[133] A Wood, G Virone, T Doan, Q Cao, L Selavo, Y Wu, L Fang, Z He, S Lin, and J Stankovic. Alarm-net: Wireless sensor networks for assisted-living and residential monitoring. Technical Report CS-2006-11, Dec 2006.

[134] Anthony D. Wood, Leo Selavo, and]John A. Stankovic. SenQ: An embedded query system for streaming data in heterogeneous interactive wireless sensor networks. In *Proceedings of DCOSS*, pages 531–543, 2008.

[135] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of SenSys*, pages 13–24, 2004.

[136] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of MobiCom*, pages 70–84, 2001.

[137] Yaling Yang and R. Kravets. Distributed QoS guarantees for realtime traffic in ad hoc networks. *Proceedings of SECON*, pages 118–127, 2004.

[138] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceeding of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1567– 1576, 2002.

[139] Kathryn Zeitz and Helen McCutcheon. Policies that drive the nursing practice of postoperative observations. *International journal of nursing studies*, 39(8):831–9, Oct 2002.

[140] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of SenSys*, pages 1–13, 2003.

[141] R. Zheng and R. Kravets. On-demand power management for ad hoc networks. *Proceedings of INFOCOM*, 1:481–491, 2003.

[142] Rong Zheng, Jennifer C. Hou, and Lui Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of MobiHoc*, pages 35–45, 2003.

[143] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher. RID: radio interference detection in wireless sensor networks. *Proceedings of INFOCOM*, 2:891–901 vol. 2, 2005.

[144] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of MobiSys*, pages 125–138, 2004.

[145] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Models and solutions for radio irregularity in wireless sensor networks. *ACM Transactions on Sensor Networks*, 2006.

[146] Hua Zhu, Ming Li, I. Chlamtac, and B. Prabhakaran. A survey of quality of service in ieee 802.11 networks. In *IEEE Wireless Communications*, 2004.

[147] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. *Proceedings of SECON*, pages 517–526, 2004.

[148] Marco Zuniga and Krishnamachari Bhaskar. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 2007.