1-1-2011

Practical and Robust Power Management for Wireless Sensor Networks

Gregory Hackmann
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Dissertation Examination Committee:
Chenyang Lu, Chair
Thomas C. Bailey
Yixin Chen
Shirley Dyke
Chris Gill
Jon Turner

Practical and Robust Power Management for Wireless Sensor Networks

by

Gregory W. Hackmann

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

DOCTOR OF PHILOSOPHY

August 2011
Saint Louis, Missouri

ABSTRACT OF THE DISSERTATION

Practical and Robust Power Management for Wireless Sensor Networks

by

Gregory W. Hackmann

Doctor of Philosophy in Computer Science

Washington University in St. Louis, August 2011

Research Advisor: Professor Chenyang Lu

Wireless Sensor Networks (WSNs) consist of tens or hundreds of small, inexpensive computers equipped with sensors and wireless communication capabilities. Because WSNs can be deployed without fixed infrastructure, they promise to enable sensing applications in environments where installing such infrastructure is not feasible. However, the lack of fixed infrastructure also presents a key challenge for application developers: sensor nodes must often operate for months or years at a time from fixed or limited energy sources. The focus of this dissertation is on reusable power management techniques designed to facilitate sensor network developers in achieving their systems' required lifetimes.

Broadly speaking, power management techniques fall into two categories. Many power management protocols developed within the WSN community target specific hardware subsystems in isolation, such as sensor or radio hardware. The first part of this dissertation describes the Adaptive and Robust Topology control protocol (ART), a representative hardware-specific technique for conserving energy used by packet transmissions.

In addition to these single-subsystem approaches, many applications can benefit greatly from holistic power management techniques that jointly consider the sensing, computation, and communication costs of potential application configurations. The second part of this dissertation extends this holistic power management approach to two families of structural health monitoring applications. By applying a partially-decentralized architecture, the cost of collecting vibration data for analysis at a centralized base station is greatly reduced.

Finally, the last part of this dissertation discusses work toward a system for clinical early warning and intervention. The feasibility of this approach is demonstrated through preliminary study of an early warning component based on historical clinical data. An ongoing clinical trial of a real-time monitoring component also provides important guidelines for future clinical deployments based on WSNs.

Acknowledgments

The work discussed in Chapters 2, 3, and 4 is supported by NSF NeTS-NOSS Grant CNS-0627126. The work discussed in Chapters 2 and 4 is additionally supported by NSF CRI Grant CNS-0708460. I would like to thank Prof. B.F. Spencer and Shin Ae Jang for the use of the truss for our experiments and all of the valuable assistance provided.

The work discussed in Chapter 5 was made possible by Grant Number UL1 RR024992 from the National Center for Research Resources (NCRR), part of the National Institutes of Health (NIH) and NIH Roadmap for Medical Research. Its contents are solely the responsibility of the author. Additional funding was provided by the Barnes-Jewish Hospital Foundation and NSF through NeTS-NOSS Grant CNS-0627126, CRI Grant CNS-0708460, and CPS Grant CNS-1035773.

This dissertation would not have been possible without the support and collaboration of my many colleagues. I would like to thank my advisor Chenyang Lu for his guidance throughout my academic career, and my committee members for their feedback on my dissertation research. I would also like to thank Nestor Castaneda, Minmin Chen, Octav Chipara, Rahav Dor, Weijun Guo, Marin Kollef, Fei Sun, and Guirong Yan, who all played an important part in the work described here.

Gregory W. Hackmann

Washington University in Saint Louis
August 2011

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) consist of tens or hundreds of small, inexpensive computers equipped with sensors and wireless communication capabilities. Because WSNs can be deployed without fixed infrastructure, they promise to enable sensing applications in environments where installing such infrastructure is not feasible. Examples of early successful WSN deployments include monitoring environmental conditions on the Great Duck Island [56], tracking the migration of Kenyan zebras [96], and monitoring the structural health of the historical Torre Aquila tower [16].

However, the lack of fixed infrastructure also presents a key challenge for application developers: sensor nodes must often operate for months or years at a time from fixed or limited energy sources. With all of the sensors' hardware components activated all of the time, a typical node will have a lifetime of less than a week when powered by AA batteries. Energy consumption has proved a significant challenge in real deployments even when the nodes are powered by renewable energy sources [96] or large battery arrays [42].

The main contribution of this dissertation is the development of reusable power management techniques which will facilitate other sensor network developers in achieving their systems' required lifetimes. Broadly speaking, these techniques fall into two categories. First, power management techniques may be applied to specific hardware subsystems; these techniques target a narrow portion of the application stack, but are generally applicable across many classes of application. Examples of hardware-specific techniques include low-power MAC layers [61, 93] and hardware drivers with integrated energy management [44]. In addition to these single-subsystem approaches, many applications can benefit greatly from *holistic* power management techniques

that jointly consider the sensing, computation, and communication costs of potential application configurations. Such techniques can have a profound effect on the efficiency of a class of applications, but may have limited applicability to others.

In this dissertation, I discuss my work on four WSN projects, spanning both classes of power management techniques. Chapter 2 describes the Adaptive and Robust Topology control (ART) algorithm, which aims to conserve energy at the network stack across a variety of applications. Chapter 3 discusses the application of holistic power management techniques to structural health monitoring applications, specifically focusing on the *Damage Localization Assurance Criterion* (DLAC) algorithm. I then discuss an evolution of this approach, using a family of numerical techniques known as *flexibility-based* damage identification, in Chapter 4. Chapter 5 describes early steps in an ongoing effort toward applying energy-efficient WSNs to clinical early warning and real-time event detection. Finally, I conclude in Chapter 6.

Chapter 2

Robust Topology Control for Indoor Wireless Sensor Networks

Topology control is an example of an effective hardware-specific technique for conserving energy in low-power wireless networks. Topology control aims to reduce power consumption and channel contention in wireless sensor networks by adjusting the radio hardware's transmission power according to wireless link characteristics. However, topology control for wireless sensor networks faces significant challenges, especially in indoor environments where wireless characteristics are extremely complex and dynamic. Wireless links have highly irregular and probabilistic properties. Furthermore, link quality can vary significantly over time, especially in indoor environments due to human activity and multi-path effects. Topology control algorithms must therefore deal with changes in link quality at different power levels using online measurements. To facilitate efficient link profiling, commodity radios [78, 79] commonly provide instantaneous link quality indicators such as the Receiver Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI). Unfortunately, the correlation of these indicators with link quality (i.e., packet reception rate (PRR)) is highly sensitive to the environment. For example, RSSI is shown to have a good correlation to PRR in some outdoor and indoor environments [53, 76], while other studies indicate the opposite [72]. A practical topology control algorithm must therefore be robust against environmental and workload changes while introducing minimal communication, processing, and memory overhead.

This chapter presents an extensive empirical study performed in an office building and provides key insights on the design of robust topology control algorithms in such environments. Based on the results of this study, we present the *Adaptive and Robust Topology control (ART)* algorithm, a practical topology control protocol algorithm

designed for complex and dynamic environments. ART has the following salient features. (1) ART is designed to be robust; design assumptions are minimized and validated through empirical studies. (2) ART is an adaptive topology control protocol that adapts the transmission power in response to variations in link quality triggered by either environmental changes (e.g., changes in signal or noise levels) or varying degrees of network contention. (3) ART is efficient, introducing zero communication overhead for applications which already use packet acknowledgements.

We have implemented ART in a topology layer between the MAC and routing layers in TinyOS, adding only 392 bytes of RAM and 1.5 kilobytes of ROM and requiring minimal changes to upper-layer routing protocols. We assess the performance of ART on a 28-node indoor testbed through micro- and macrobenchmarks. The microbenchmarks demonstrate that ART can lower the energy consumption of individual links by an average of 15% with no loss in link quality. The macrobenchmarks show that ART's performance meets that of a representative topology control algorithm, PCBL, without introducing PCBL's bootstrapping cost. We also show that ART can improve energy efficiency by up to 40% under heavy contention.

2.1 Related Work

In this section, we describe existing studies into the impact of various environmental and spatial factors on the performance of wireless sensor network links. We also discuss state-of-the-art topology control algorithms that attempt to select the proper power setting for wireless links in the face of the complex behavior observed in these studies.

2.1.1 Empirical Link Studies

A significant number of existing link quality studies [49, 67, 76, 87, 98] evaluate link performance at a single, fixed power setting. More closely related to the work in this chapter are empirical link studies which explore the properties of wireless links at varying power levels. [30,97] observe that radio ranges are highly irregular and do not fit circular radio models in practice. [72] finds a temporal impact on link quality with the Chipcon CC1000 radio [79] in an indoor testbed environment and notes that some

middling-quality links can be converted to good-quality links with small changes in transmission power. [53] notes a strong correlation between RSSI and PRR on the Chipcon CC2420 radio [78], independent of time or transmission power, in three different environments; it also shows a strong correlation between transmission power and RSSI which varies across links and across time.

The link quality study discussed in Section 2.2 builds on these existing studies and is complementary to this work. These experiments are carried out on a complex indoor testbed of motes equipped with 802.15.4-compliant Chipcon CC2420 radios; previous studies consider older, proprietary radios [30,72,97] or were carried out in a simplified indoor environment [53]. The study also provides new insights into several areas of interest to topology control algorithms, such as the impact of transmission power on contention; whether link indicators are robust indoors; and whether links which are high-quality over the short term remain high-quality over the long term.

2.1.2 Topology Control Schemes

We discuss here a number of existing topology control algorithms. We begin by examining a class of theoretical algorithms that have been evaluated only in software simulations. We will then describe two state-of-the-art topology control algorithms, which are based on extensive link quality studies and have been deployed on real sensor networks. Throughout this section, we highlight several key assumptions made by these algorithms; Section 2.2 evaluates the robustness of these assumptions as part of the empirical study.

Theoretical Model-Based Algorithms

Traditionally, topology control schemes have been built on simplified theoretical radio models (such as graph-based connectivity) and tested in simulation environments. However, these traditional topology control schemes are not always appropriate for wireless sensor networks. These topology schemes make simplifying assumptions such as circular radio range [51, 52] or uniform node distribution [9] which are unrealistic in many wireless sensor network applications. Moreover, a typical wireless sensor node has limited computational power and storage capacity, which mandate topology control algorithms with low processor overhead and memory consumption.

A more recent trend in theory-based topology control is to quantify the interference of a network graph and explicitly consider this metric when selecting a network topology. Theoretical models that consider interference represent a more accurate view of real-world wireless sensor network environments than those that do not. Nevertheless, existing interference-based algorithms still make unrealistic simplifying assumptions such as circular radio range [14], global knowledge [10], or the ability to perform complex computations online [33]. These assumptions limit the applicability of these algorithms for real sensor network deployments.

LMST [52] is a representative theoretical model-based algorithm which is specifically designed to fit the communication and computational constraints of wireless sensor networks. LMST computes a reduced-power network topology by constructing a minimum spanning tree over the network in a fully-distributed fashion. The transmission power of each link in this topology is shrunk according to the observed path loss. When computing the network's MST, LMST assigns a cost to each link proportional to its physical length. However, existing studies have found a much more complex relationship between link length and link quality [30, 72, 97]. LMST also requires that all nodes know the physical distance to their neighbors in order to operate in a truly decentralized fashion.

PCBL

PCBL [72] is one of the first topology control algorithms to be deployed in a real-world sensor network testbed. Using extensive empirical data collected from an indoor testbed of PC104 motes, the authors observe that high-quality links also tend to be highly stable. Specifically, they note that links with a PRR above 98% during a seven-day experiment had a standard deviation of 2.2%, while links with a PRR above 90% had a standard deviation of 19.8%. Based on these observations, the authors proposed that PCBL maintain two separate bounds on link quality. Links which fall below the lower PRR bound at all power levels are considered unreliable and blacklisted (i.e., never used for transmissions). Links which achieve the upper PRR bound at some power setting are considered highly reliable, and their power is shrunk to the lowest such setting. Finally, links which fall between these two extremes are considered reliable enough to use, but only at maximum power.

It is impractically expensive to maintain comprehensive PRR data over all power levels at runtime, since this would require PCBL to continuously probe a node's neighbors at all power levels. Instead, the basic PCBL algorithm approximates this behavior by extensively probing links once at all power settings and then freezing this link quality data. The intuition behind this approximation is that high-quality links are also highly stable, so PCBL will inherently favor links which are resilient to changes in network conditions.

[72] suggests that PCBL's runtime overhead could be lowered by initially collecting link statistics at maximum power, allowing the routing layer to bootstrap and select its neighbors, and then only tuning the transmission power over those links actually used for routing. This lowered overhead would allow PCBL to bootstrap more frequently and hence be even more resilient against link quality fluctuations. However, this approach would conflict with highly-dynamic routing engines like TinyOS's Collection Tree Protocol [34] which continuously send beacons during the application's lifetime to discover less-expensive routes.

ATPC

ATPC [53] is designed to avoid costly link probing by using an instantaneous link quality metric as a proxy for PRR. The authors gathered RSSI, LQI, and PRR statistics in three different environments: a parking lot, a grassy field, and an office building. They discovered a strong correlation between RSSI and PRR (and between LQI and PRR) along a monotonically-increasing curve. The shape of this curve varied for each environment but was consistent across all links, power settings, and times within a given environment. Once the authors collected enough data offline to construct this curve, they were able to convert a lower-bound on PRR into a corresponding lower bound on RSSI or LQI.

The authors also noted a linear correlation between transmission power level and RSSI/LQI readings at the receiver. Unlike the RSSI-to-PRR curve, this parameters of this line varied across links and over time. ATPC estimates the slope and Y-intercept of this line at runtime for each link and dynamically adjusts this model using a closed feedback loop. Using this model, ATPC selects the proper transmission power to achieve the necessary lower bound on RSSI and, by proxy, the lower bound on PRR.

2.2 Empirical Study

This section present an empirical study which considers four questions at the core of topology control algorithms: (1) is topology control beneficial?; (2) what is the impact of transmission power on contention?; (3) is it necessary to dynamically adapt transmission power online?; and (4) are instantaneous indicators of link quality (such as RSSI and LQI) robust in indoor environments? In answering these questions, we provide guidelines for the development of topology control algorithms, which we apply to the design of the ART algorithm in Section 2.3.

The results presented in this section are complementary to the empirical power control studies presented in [53,72]. The empirical results presented in [72] were obtained using the CC1000 radio platform; the experiments in this section are performed on Chipcon CC2420 radios, which comply to the IEEE 802.15.4 physical layer specification. The two radio platforms are significantly different [78, 79]: they operate in different frequency domains and use different modulation schemes. The empirical results in [53] were also obtained using the CC2420 radio; however, they considered only simple network layouts where nodes have line-of-sight communication. In contrast, the deployment used in this study spans a floor of an entire building; a significant number of links are also formed between nodes without line-of-sight.

The empirical study presented here validates many of the findings of previous studies. Moreover, we also provide important new insight into the impact of transmission power on contention. In particular, we find that some assumptions underlying the design of PCBL and ATPC do not hold on our testbed. These discrepancies are caused by the different radio platforms and environments in which the experiments were performed. These observations served as the foundation for developing a topology control algorithm that is *robust* in different indoor environments.

2.2.1 Experimental Setup

All experiments are performed on a testbed consisting of 28 TelosB motes equipped with CC2420 radios using Tiny-OS 2.x's default CSMA/CA MAC layer. Each node is connected to a central server using a wired USB and Ethernet backbone. This backbone is used as a back-channel to issue commands to the motes and collect experimental results without interfering with ongoing wireless transmissions.

The CC2420 radio chip can be programmed to operate at 8 different power levels[1] with output power ranging from −25 dBm to 0 dBm and current consumption ranging from 8.5 mA to 17.4 mA [78]. The CC2420 radio also provides an RSSI reading and LQI reading embedded in the metadata of all incoming packets. The RSSI reading represents a sampling of the signal strength (transmission + background noise) taken at the beginning of the packet reception, while the LQI reading represents the average symbol correlation value over the packet's first eight symbols. In order to estimate background noise, applications may sample the CC2420's RSSI register when the radio is idle. The CC2420 may be programmed to operate on different frequencies; all experiments here are performed on a channel that does not overlap with the building's 802.11g network.

2.2.2 Is Topology Control Beneficial?

Topology control is an attractive mechanism because it can simultaneously improve energy efficiency and network performance. Here, we evaluate the potential benefits of topology control in our indoor testbed.

To isolate the impact of topology control on link quality and energy savings, we performed an experiment in which there is no network contention. Each node broadcasts 50 packets while its neighbors record the sequence number, RSSI reading, and LQI reading of all packets that they receive. The node repeats this procedure at each of the CC2420's eight discrete power levels for a total of 400 packets. After a node completes sending its 400 packets, the next sending node is selected in a round-robin fashion. This cycle repeated for 24 hours, giving each node 49 rounds to transmit 400 packets each.

During the course of the experiment, at least one successful packet transmission was recorded on 524 of the network's 756 possible unidirectional links. Figure 2.1 shows the network topology during this experiment at three different power levels, where line thickness is proportional to the packet reception ratio (PRR). The topology at maximum power (0 dBm) and medium power (−5 dBm) are fairly similar, both in terms of connectivity and link quality. In contrast, the minimum power (−25 dBm) topology has partitioned the network into clusters of mostly high-quality links. These

[1]Though the CC2420's transmission power register can be set between 0–31, the CC2420 datasheet only defines output power information for 8 of these 32 settings.

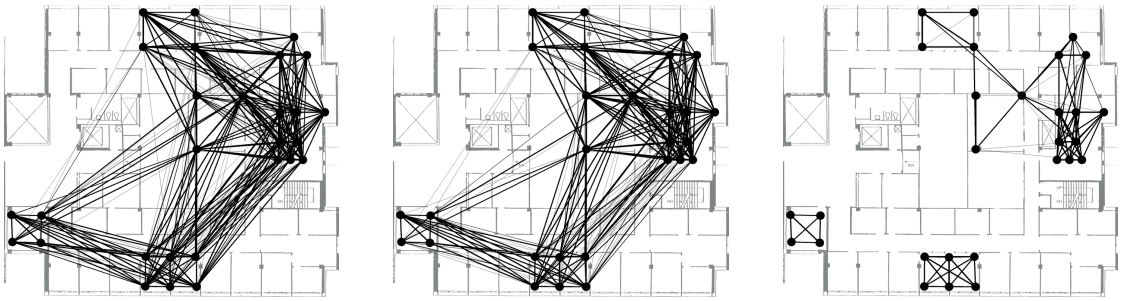
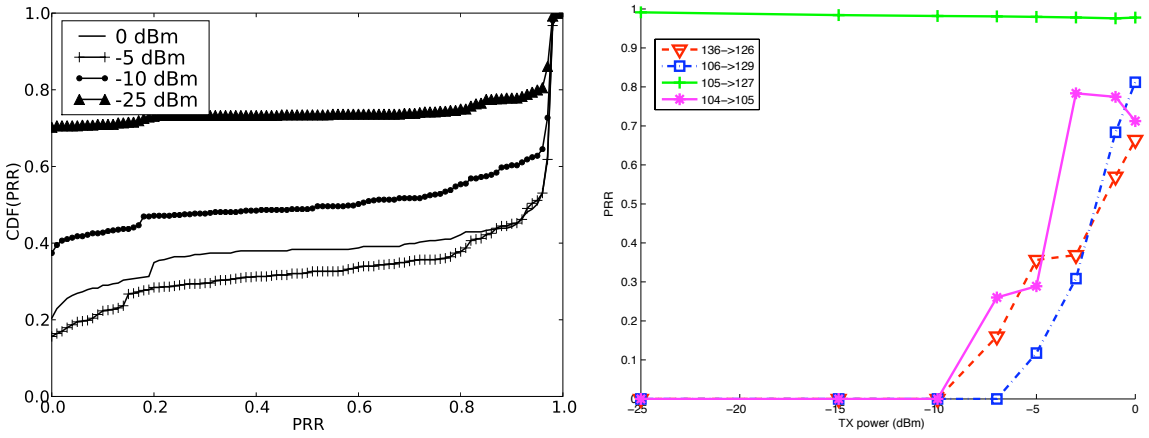


Figure 2.1: The testbed topology when transmitting at 0 dBm, −5 dBm, and −25 dBm

figures highlight the potential benefits of assigning non-uniform transmission powers to different nodes: it is sufficient to transmit at −25 dBm to reach nearby nodes, but other links require higher transmission powers to achieve connectivity. This confirms experiments carried out in [72] and [53] showing that uniform power settings across the network lead to non-uniform behavior.



(a) PRR distribution

(b) Impact of transmission power on PRR

Figure 2.2: A significant fraction of poor quality links may be transformed into high quality links through topology control

To better understand the benefits of tuning the transmission power, we computed the PRR of each link during the entire benchmark run; the CDF of link PRRs is shown Figure 2.2(a). Changing the transmission power can affect a large fraction of the links in the testbed: for example, 368 links (70.2%) have a PRR of 0 at −25 dBm, compared to 82 (15.6%) at −5 dBm. Figure 2.2(b) shows a similarly dramatic effect, where three links selected from our testbed go from unusable at -10 dBm to medium quality at -5 dBm. This confirms the results in [72] which indicate that transmission

power can transform a significant number of poor quality links into good quality links. We also note that a slightly higher proportion of links have poor link quality at 0 dBm than at -5 dBm in Figure 2.2(a). A handful of nodes performed worse when transmitting at 0 dBm than at lower power settings, which we believe is caused by multi-path effects that are more pronounced at maximum power. This phenomenon may also be seen in Figure 2.2(b) for link $104 \rightarrow 105$.

Similarly, reducing a link's transmission power can result in significant energy savings. To quantify these savings, we inspected the traces collected during the previous experiment. For each link, we computed the PRR for each round using the max-power data, and then selected the lowest power level for each round that had no degradation in PRR compared to max-power (i.e., the power setting that a topology control algorithm with perfect knowledge *would have* picked). When the maximum transmission power is used, a node would draw 17.4 mA, compared to an average current draw of 11.4 mA under an ideal power assignment.

Insight 1: *Transmission power should be set on a per-link basis to improve link quality and save energy.*

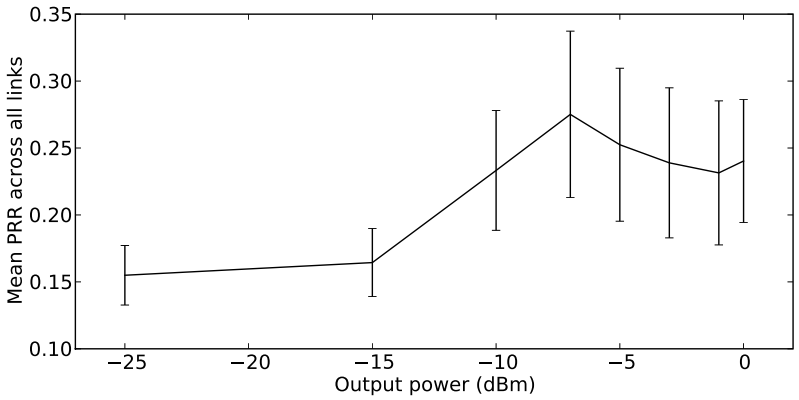


Figure 2.3: Effect of transmission power on contention

2.2.3 What is the Impact of Transmission Power on Contention?

While increasing transmission power improves the quality of *individual* links, its also may result in increased contention. To understand the impact of transmission power on contention, we performed the following experiment. Ten links (selected at random

from the testbed) simultaneously transmit packets as fast as possible for 24 hours. Each node transmits 200 packets, after which the power level is changed. Figure 2.3 plots the average PRR over all links as transmission power is increased from -25 dBm to 0 dBm. Increasing the transmission power up to -7 dBm has a positive effect on link quality; however, increasing the power further results in decreased PRR. This happens because the benefits of increased transmission power are offset by higher contention, increasing packet collisions and decreasing throughput. The data indicates that power control is an effective mechanisms for controlling the degree of network contention. Moreover, robust topology control cannot be performed without accounting for this link quality/contention tradeoff.

Insight 2: *Robust topology control algorithms must avoid increasing contention under heavy network load.*

2.2.4 Is Dynamic Power Adaptation Necessary?

One important consideration for topology control protocol design is the rate at which their power decisions need to be re-evaluated. If the rate of change in link quality is sufficiently low, then it is feasible to make infrequent decisions that incur high communication or computational overhead. To address this question, we ran a long-term experiment on a few links to determine the time-scale of link variations. The setup for this experiment is identical to that in Section 2.2.2, except it was carried out over only 3 links with 100 packets per power level per round. By reducing the number of links profiled and increasing the number of packets per link, the experimental traces capture a fine-grained view of how link quality varies over time. We remind the reader that we select a radio frequency for these experiments with little background noise. However, the nodes also sample 50 signal strength readings in the beginning of each round when no node is transmitting, to validate that the background noise does not vary significantly over the duration of the experiment.

Figure 2.4 shows the PRR, RSSI, and background noise for one of the sampled links; the other measured links had similar results. Figure 2.4 indicates a correlation between RSSI and human activity: during work hours, from 8:00 to 18:00, there is a significant reduction in RSSI along with an increase in its variation; in contrast, during the night, the link is significantly more stable.

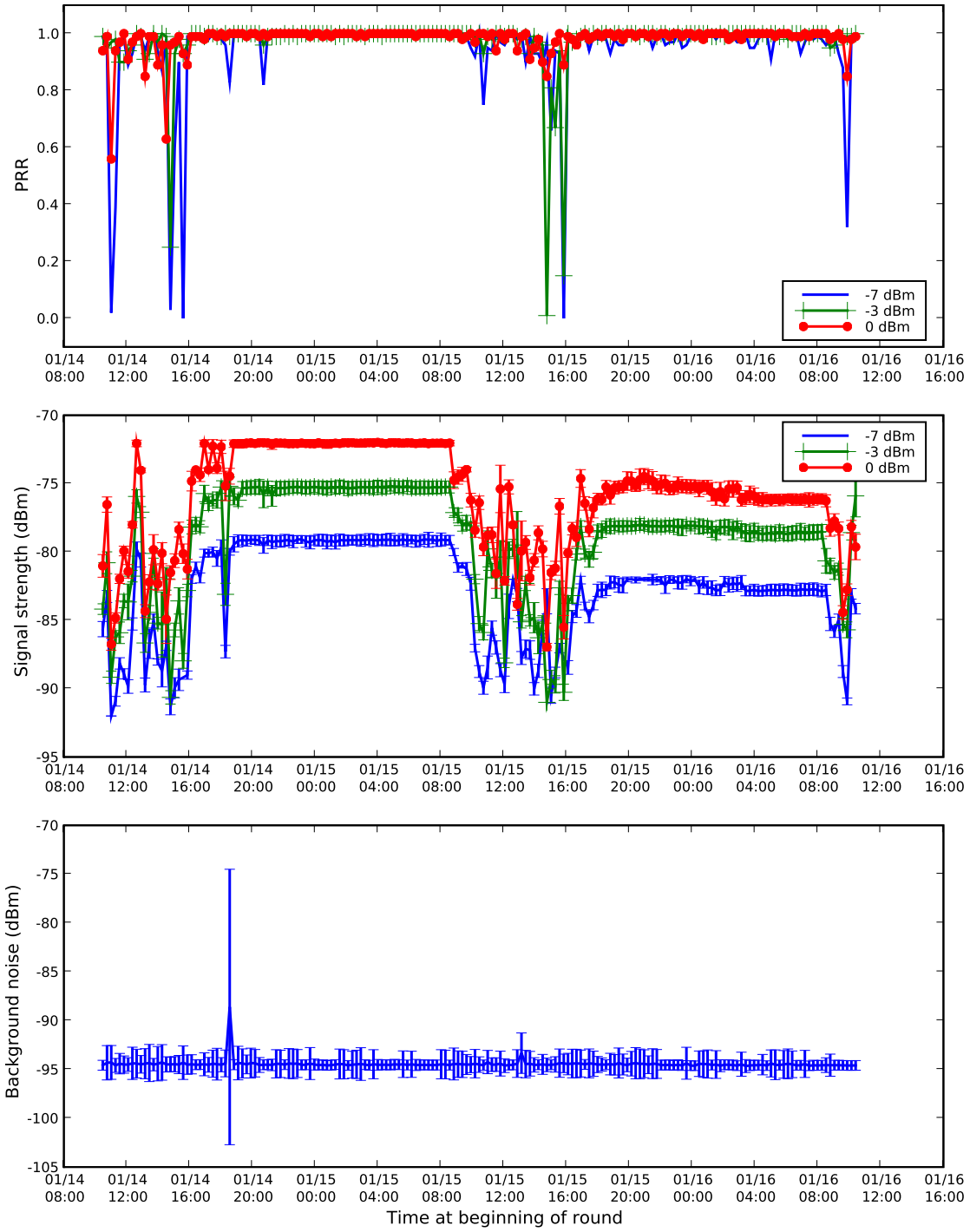


Figure 2.4: The PRR, mean signal strength, and background noise collected over link $110 \rightarrow 139$

A similar correlation may be observed for PRR, which is more pronounced for lower transmission powers. This correlation is expected because even small noise variations may cause packets transmitted at lower power levels to be corrupted. The trace shows that in order to maximize energy savings, the transmission power must be tuned dynamically based on environmental conditions.

Apart from one outlying data point, the background noise on the wireless channel is stable during the entire benchmark run, suggesting that increased activity on the wireless spectrum during the workday was not responsible for this cyclical link quality fluctuation. The sharp variations in link quality may be attributed to people walking around the building during the daytime; similar results were observed on the CC1000 radio [72]. These results indicate that topology control schemes must frequently adapt link transmission power over time in order to avoid significant variations in link quality.

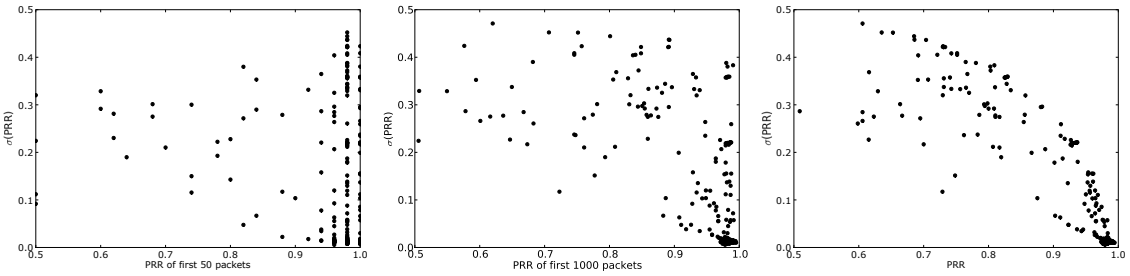


Figure 2.5: The relationship between overall PRR and standard deviation in PRR

An important characteristic of high-quality links is that they have high PRR and low standard deviation. This assumption has been validated on long-term experiments on the CC1000 radio [17, 72]. Protocol developers use this assumption to argue in favor of performing a bootstrapping phase in which such high quality links are identified. Unfortunately, based on the data collected in Section 2.2.2, a strong correlation cannot be established between PRR and standard deviation without collecting an impractically large amount of PRR data. Figure 2.5 plots the relationship between each link's overall standard deviation in PRR (i.e., its actual stability over the entire benchmark) against the PRR calculated by looking at the first data round (50 packets/link), 12 hours' worth of link data (1000 packets/link), and the entire benchmark dataset (1950 packets/link). Links with an overall PRR of 98% or higher during the full 24-hour dataset indeed have low standard deviation (right). However, looking at links with a 98% PRR within the *first* round of data (left) results in a 6.8-fold increase in standard deviation. Even selecting links with $\geq$ 98% PRR over 12 hours'

of data (center) would result in a 3.2-fold increase in standard deviation compared to the full dataset. This indicates that, in some environments, even many hours' worth of bootstrapping data is insufficient to properly predict a link's long-term behavior. PCBL [72] suffers from this design pitfall, because a short bootstrapping phase is used to predict the long-term link quality.

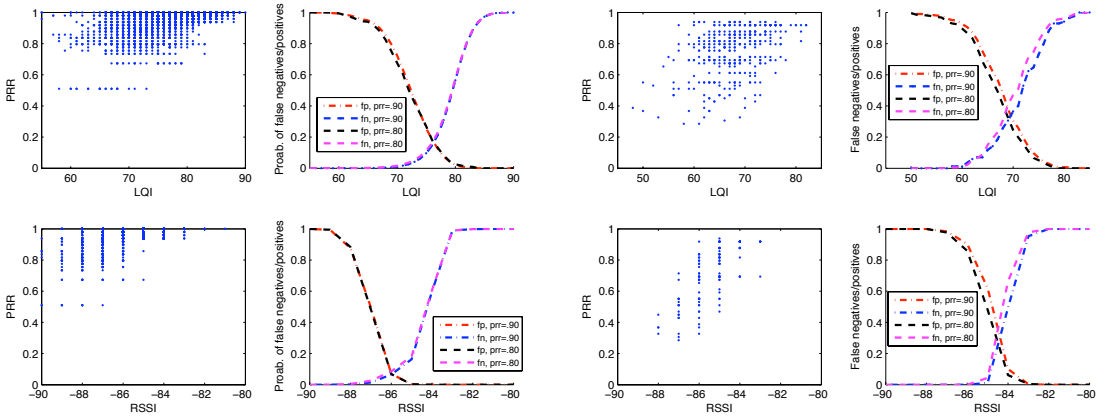
Insight 3: *Robust topology control algorithms must adapt their transmission power in order to maintain good link quality and save energy.*

2.2.5 Are Link Indicators Robust Indoors?

Commodity radios generally provide per-packet link quality indicators, such RSSI and LQI. Because these metrics are instantaneous and inexpensive to collect, they are an attractive proxy for more expensive link quality indicators such as PRR. The relative benefits of these metrics are still a subject of debate in sensor network community: for example, [76] advocates measuring link quality using RSSI over LQI, while [53] reports that both RSSI and LQI are good indicators of link quality. This subsection present an empirical study designed to understand if either RSSI or LQI are *always* good indicators of link quality in complex indoor environments. To this end, we performed an experiment which transmitted 50 packets over each pairwise link in our testbed at maximum power. Each time a packet was received, the recipient logged the packet's sequence number, RSSI reading, and LQI reading. By restricting experiments to a single power level, 672 rounds of data were collected over a period of 32 hours.

If there exists some critical RSSI or LQI threshold which separates "good" (high PRR) links from "bad" (low PRR) links, then these inexpensive metrics could be used as good indicators of PRR. Using several different PRR thresholds to define "good" links, we aimed to find these critical RSSI and LQI thresholds in the experimental collected data collected above. These critical thresholds should represent the best compromise between false positives (i.e., links above the LQI/RSSI threshold but below the PRR threshold) and false negatives (i.e., links above the PRR threshold but below the LQI/RSSI threshold). For the purpose of brevity, the results for two links are presented here at PRR thresholds of 80% and 90%.

Figure 2.6(a) shows the results for link $106 \rightarrow 129$. Each point in the scatter plots represents the relationship between mean RSSI/LQI and PRR for one 50-packet round.



(a) Link 106 → 129

(b) Link 104 → 105

Figure 2.6: Quality of RSSI and LQI as instantaneous indicators of link quality

The graphs seem to indicate a correlation between the instantaneous link estimators and PRR. When setting the PRR threshold to 90%, the best trade-off between false positive and false negative rates is at LQI = 76. At this threshold, the false positive and false negative rates are at 18% and 16% respectively. RSSI yields similar results, with a false positive rate of 6% and false negative rate of 8% at the critical RSSI threshold of 86 dBm. The results when reducing the PRR threshold to 80% are similar. These results indicate that both LQI and RSSI are good indicators of this link's quality.

Figure 2.6(b) shows the results for link 104 → 105. The scatter graphs indicate that the correlation between link indicators and PRR is worse than that observed on the previous link. Indeed, when setting the PRR threshold to 90%, the optimal LQI threshold of 70 has a false positive rate of 30% and a false negative rate of 36%. RSSI performs even more erratically, with the optimal threshold being at either -85 dBm or -84 dBm. At an RSSI threshold of -85 dBm, the false positive rate is 4% while the false negative rate is 62%; increasing the RSSI threshold to -84 dBm causes these rates to jump to 66% and 6% respectively. This sharp transition indicates that RSSI would be an unstable estimator of this link's quality, while LQI would have a significant fraction of false negatives and positives. Similar behavior is observed with a PRR threshold of 80%.

This set of experiments demonstrate that, although there are links for which LQI and RSSI are good link quality indicators, there are others for which they are both poor

indicators. Accordingly, neither LQI nor RSSI may be used for developing *robust* topology control algorithms.

ATPC [53] relies on RSSI as an indicator of link quality. Since RSSI is not *always* a good indicator of high quality links, we do not expect ATPC to be sufficiently robust for operating in all indoor environments.

Insight 4: *Instantaneous LQI and RSSI are not robust estimators of link quality in all environments.*

2.3 The ART Algorithm

This section presents the design of a new topology control algorithm, *Adaptive and Robust Topology control (ART)*. ART is designed based on the key observations in the previous section and has the following salient features. (1) ART is designed to be a *robust* topology control algorithm: it does not use indirect measurements of link quality because they are not sufficiently robust in different indoor environments. (2) ART is an *adaptive* algorithm, in that it changes the transmission power of a link based on its observed PRR. Moreover, ART employs a lightweight adaptation mechanism and does not employ prolonged bootstrapping phase for link profiling. (3) ART can dynamically adapt the transmission power in response to high channel contention. (4) ART is specifically designed to be *efficient*, so that it can be realistically deployed on memory-constrained wireless sensor platforms with low runtime overhead.

2.3.1 ART Algorithm Description

ART individually tunes the transmission power over each of a node's outgoing links. A link is initially set to transmit at its maximum power. ART monitors all outgoing packet transmissions and keeps a record of whether each transmission failed or succeeded in a sliding window of size w. While the window is filling, a link is said to be "initializing". When the sliding window is full, ART compares the number of recorded transmission failures to two thresholds d and d', where $d > d'$. The link remains in this "steady" state as long as the number of failures is between these two thresholds.

If the recorded number of failures is above d, then ART adjusts the link power to improve its quality. ART may raise the link's transmission power to improve its quality under low contention, but may lower its transmission power under high contention. As detailed in Section 2.3.3, ART uses a simple gradient-based mechanism to detect high contention based on recent history. After selecting a new power setting, ART will flush the transmission window and re-enter the initializing state.

If the number of failures is below d', then ART will enter a "trial" state where it temporarily lowers the power by one level. If the link experiences d' more transmission failures at any time while in the trial state, then it returns to the previous power level, flushes its transmission window, and goes back to the initializing state. If the link's window fills with fewer than d' recorded transmission failures, then the new power setting is made permanent and the link goes back to the steady state.

This algorithm has several salient features worth emphasizing. First, it makes its decisions by monitoring traffic already being generated by the upper layers, and therefore introduces no communication overhead apart from packet acknowledgements[2]. Second, the largest component of its memory overhead is a sliding window of one bit per entry, which in practice can be as small as 20 to 50 entries per link. Third, ART makes no computationally-complex decisions, and hence can be implemented with minimal ROM and CPU overhead. Finally, ART does not always assume that increasing transmission power will improve link quality, since this is not always the case under high contention and interference (see Section 2.2.3).

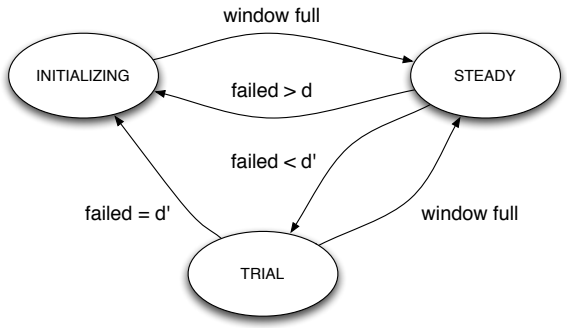


Figure 2.7: ART state transition diagram

Figure 2.7 summarizes the ART algorithm as a state diagram. We will now discuss in further detail the intuition behind this algorithm.

[2]Because packet acknowledgements are needed for reliable data transmission and part of many routing protocols, in practice ART will often receive these acknowledgements "for free".

2.3.2 Link Quality Thresholds

The most straightforward indicator of a link's quality is its PRR. ART is therefore designed to lower the link's power while still maintaining an application-specified target PRR. Consider an application that specifies a target PRR of p. ART can inexpensively track the link's recent behavior by creating a sliding window of w bits, where each bit represents a single transmission and indicates whether it was ACKed by the recipient. A link meets a target PRR p if there are $d = w \cdot (1-p)$ or fewer failed transmissions in its window at any given time. Each link's power should therefore be tuned to the lowest power that can keep d or fewer failures in the link's window.

However, a link's transmission power cannot necessarily be lowered each time it is within its upper-bound on failures. Because of the bimodal relationship between transmission power and PRR observed in Section 2.2.2, this might result in an actual PRR much lower than p. After the link changes power, it must flush its packet window to reflect the potential change in link quality. The application must therefore transmit an entire window of w packets over the link before ART can detect that its link quality has degraded. If the link's previous power was at the critical threshold between a high-quality link and a low-quality link, there may be more than d failures at its new power state. In the worst case, the link may alternate between two power states in which $d-1$ transmissions fail in one window, followed by all w in the next window.

ART therefore incorporates two policies to address this problem. First, an entire window does not need to be filled to detect link failure. As soon as there are d delivery failures in the window, it is impossible to meet the target PRR p once the window is full. Thus, links are first moved into a "trial" state, where they transmit packets at a lower power setting but immediately return to a higher power level if too many failures are detected. Links which successfully pass the trial with a full window are moved back into the "steady" state at the reduced power.

Second, we create another PRR threshold to accommodate the potential transmission failures in this "trial" state. Specifically, we want to select a second PRR threshold p' and a corresponding failure threshold $d' = w \cdot (1-p')$. We select p', d' such that a link may lose up to $d' = w \cdot (1-p')$ packets during a w-packet window, allowing it trial to a lower power; fail up to d' packets at the lower power level, forcing it back to its original power level; and still achieve a PRR of p. In the worst case, this is satisfied by

failing $d' = w \cdot (1 - p')$ out of w transmissions at the current power level, then failing all $d' = w \cdot (1 - p')$ out of the next d' at a lower power; i.e., $\frac{w \cdot (1-p') + w \cdot (1-p')}{w + w \cdot (1-p')} \leq 1 - p$. This inequality holds when $p' \leq \frac{2p}{p+1}$, and hence we choose $d' = \frac{2p}{p+1} \cdot w$.

2.3.3 Handling High Contention

Consider the case when a link fails, i.e., it falls below its PRR threshold of p by accumulating more than d transmission failures in its window. As noted in Section 2.2.3, increasing transmission power can in fact decrease overall link quality, due to contention and interference from other nodes. ART addresses this problem by maintaining a flag which indicates whether to increase or decrease transmission power on link failure. This flag is initially set so that ART responds to link failure by increasing its transmission power by one level.

Each link also maintains a one-element history recording the number of transmission failures in its window the last time that the link failed. When ART determines that a link has failed, it compares the current failure count against this history. If the current failure count is higher (i.e., increasing the power made things worse), then ART inverts this flag. In effect, this flag allows ART to track the "gradient" of its current position on the power/PRR curve without maintaining a full multi-window history.

Rather than devising a complex scheme for detecting congestion or coordinating the power decisions across nodes, ART uses this "gradient" solution due to its *simplicity* and *elegance*: (1) minimal state is required for maintaining the gradient; (2) there is no significant processing overhead; and (3) it introduces no additional communication overhead. Macrobenchmarks presented in Section 2.5 show the effectiveness of the proposed solution.

2.3.4 Handling Broadcast Traffic

Throughout this section, we have assumed that the radio is sending traffic in a strictly unicast fashion. However, broadcast traffic is frequently used in sensor network applications for disseminating information to multiple neighbors in the node's one-hop neighborhood. It is important to note that there are actually two distinct types of

broadcast traffic as far as topology control is concerned: true broadcast data, and multicast data.

Multicast data packets are those which need to be distributed to all (or some subset) of neighbors in the node's neighbor table. For example, data dissemination packets may fall under this category. For these packets, ART transmits with the maximum power setting among all the node's one hop neighbors. This policy ensures that all neighbors can receive the message, but that the node may still be able to transmit below maximum power if it has good-quality links to all of its neighbors.

True broadcast packets, on the other hand, should be sent to all one-hop neighbors *including* those that are not in the node's neighbor list. Routing-layer beacon packets are a good example of this kind of traffic: lowering the power setting to cover the known neighborhood is counter-productive, since the routing layer intends to discover neighbors that it does not already know about. ART handles this traffic by broadcasting it at maximum power.

2.3.5 Overhead Analysis

We note that ART generally introduces no communication overhead aside from packet acknowledgements, since it operates solely on data packets being sent by the upper layers. ART also has very low memory overhead: it needs a single bit to track the contention "gradient"; one byte to track the broadcast power; three bytes per link to track its state, transmission power, and last packet failure count; w bits per link to store its PRR window data; and $2 \log w$ bits per link to store its window size and position. As demonstrated by measurements in Section 2.5.1, this RAM requirement is well within the capabilities of existing sensor network hardware.

In networks with sporadic traffic patterns, there may not be enough data packets for ART to keep its sliding window up-to-date. There should be at least w packets sent within the time that it takes for links' quality to fluctuate (which is dependent on network properties). ART could be augmented to deal with low-traffic workloads by injecting beacon packets into the network when the transmission rate over a link falls below this lower bound. Note that, because ART operates below the routing and link layers, it can often leverage these layers' control packets to update its sliding window even in the absence of application-layer transmissions.

2.3.6 Energy Efficiency

As a topology control algorithm, ART is designed to minimize the transmission power of individual links. In many sensor network applications, it is important to reduce the total *energy* consumption, since it has a direct impact on sensor lifetime.

Because ART is designed to minimize the transmission power of a link without violating the user-specified PRR bound, its energy efficiency depends the user selecting an appropriate PRR threshold for their application. For example, a threshold of 50% may not be appropriate for an application which requires 100% end-to-end reliable data transmission, because ART may cause nodes to spend more energy on retransmissions than is saved by reducing the radio power. A threshold of 95% would be more appropriate for this environment, since a 5% retransmission overhead would likely be offset by similar or larger reductions in transmission power. (For example, on the CC2420 radio, even reducing the output power level from 0 dBm to -1 dBm will reduce the radio's current consumption by over 5%.) As shown in Section 2.5, ART is able to reduce transmission power with proportionally smaller drops in PRR. In addition, because of ART's contention-handling optimization, it is sometimes able to *increase* the PRR while reducing the transmission power. These benchmarks demonstrate that ART is energy-efficient in practice.

2.4 Implementation

In this section, we present our implementation of ART for the TinyOS 2.1 operating system [1]. ART is implemented on top of the component-based MAC Layer Architecture (MLA) [43]. MLA augments TinyOS's low-level radio drivers to provide the hardware-independent interfaces required by timing sensitive power management protocols. By leveraging these pre-existing interfaces, we are able to implement ART as platform-independent components within MLA. MLA also includes components that represent common MAC functionality and implements several optional power-saving MAC layers; we used a MAC layer which implements TinyOS 2.1's default CSMA/CA logic.

In this section, we discuss two major aspects of the implementation effort. First, we discuss the design of a new topology control layer on top of MLA. Second, we describe

changes to TinyOS's Collection Tree Protocol (CTP) [34] routing layer to allow it to modularly support a variety of underlying topology control schemes.

2.4.1 Interfacing with MLA

To implement ART, we used the existing MLA codebase and inserted a topology control layer into the radio driver architecture above the existing MAC layer, as shown in Figure 2.9. In keeping with the MLA design goals, we wished to design the topology control layer in a hardware-independent fashion, allowing it to be plugged into future MLA-supported radio stacks with little or no additional effort. We found that TinyOS and MLA already included platform-independent interfaces for the majority of the radio functionality needed by topology control schemes. However, there were two specific radio features for which we needed to create platform-independent hooks: adjusting the radio power and getting the signal strength of incoming packets.

To allow the topology control layer to adjust the radio power, we created the `PacketPower` interface:

```
interface PacketPower {
  async command uint8_t getPower(msg);
  async command void setPower(msg, power);

  async command uint8_t minimum();
  async command uint8_t maximum();
}
```

The `getPower()` and `setPower()` commands respectively get and set fields in the packet metadata corresponding to the power level at which the radio should transmit the packet. These commands are taken from TinyOS's `CC2420Packet` interface, where the 8-bit `power` value is mapped directly onto the format of the CC2420's 5-bit `PA_LEVEL` register. We therefore adjusted the semantics of the `PacketPower` interface to be more radio-independent. We added `minimum()` and `maximum()` commands to represent the range of the radio output power, and defined the behavior of the `getPower()` and `setPower()` commands so that all discrete values between `minimum()` and `maximum()` inclusive are mapped to radio-supported settings. As noted in Section 2.2, the CC2420 datasheet only defines the power output behavior

for 8 of the possible 32 PA_LEVEL settings: 3, 7, 11, etc. We therefore modified the CC2420 stack to present its power range to the application layer as the contiguous range 0 ...7, which it maps internally to supported PA_LEVEL settings.

The PacketQuality interface contains a single getRssi command, which returns the signal strength of an incoming packet[3]:

```
interface PacketQuality {
  async command int8_t getRssi(msg);
}
```

Like the getPower() and setPower() commands, we extracted the existing getRssi() command from TinyOS's CC2420Packet interface but modified its semantics to be radio-independent. The CC2420Packet interface directly returns the RSSI reading provided by the CC2420 radio chip, which is ∼45 dBm above the actual signal strength [78]. We redefined getRssi() to return the actual signal strength of the packet in dBm, and correspondingly modified the CC2420 stack to subtract 45 from RSSI readings provided to the application layer. While the ART algorithm does not use this functionality, other topology control algorithms (such as ATPC) require RSSI readings to select the appropriate output power level.

We also observed that the current CC2420 radio stack leaves the radio's PA_LEVEL register set according to the most recent data packet transmitted. This behavior has a subtle implication for ACK packets: they will be transmitted at whatever power setting the last data packet was transmitted at, even if the ACK is being sent to a different neighbor. Because the CC2420 radio automatically generates ACK packets in hardware, we cannot instrument the CC2420 stack to set the optimal power setting of these ACK packets according to the topology layer's decision. Instead, we reset the PA_LEVEL register to the maximum power after transmitting each data packet, so that all subsequent ACKs will be sent at max power.

2.4.2 Interfacing with CTP

The tree-based CTP routing protocol is the default routing protocol in TinyOS 2.x. CTP designates one or more nodes in the network as sink nodes. All other nodes in

[3] PacketQuality does not include a corresponding getLqi() command since LQI only applies to physical layers based on the 802.15.4 specification.

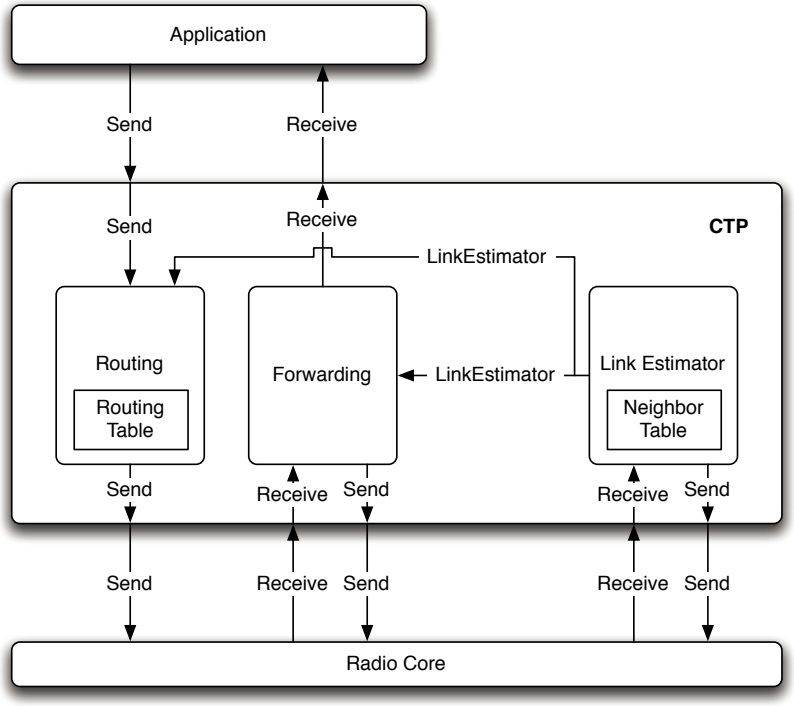


Figure 2.8: CTP's original organization and architecture

the network recursively form routing trees which are each rooted at one of these sink nodes. Nodes periodically broadcast beacon packets which serve two purposes. First, they contain a sequence number field which TinyOS's link estimator component uses to compute the Estimated Transmission Count (ETX, roughly $\frac{1}{PRR}$) to each node's one-hop neighbors. Second, nodes embed in these advertisements an estimate of the total cost (initially 0 for sink nodes and ∞ for all other nodes) of routing a data packet to the sink through them. Non-sink nodes then select a parent on a routing tree by collecting their advertised routing costs, adding their one-hop ETX, and selecting the neighbor with the lowest total routing cost. Because CTP sends these beacons periodically, nodes can dynamically change their parents as link quality fluctuates.

The topology control layer is largely agnostic to the routing and application layers sitting on top of it: it only requires an external neighbor table for storing its own link quality data at runtime. We discovered that CTP's default implementation is poorly-suited to allow other components to embed data in its neighbor table. As shown in Figure 2.8, TinyOS does not provide a single shared neighbor table component. Instead, TinyOS's link estimator component (which computes the ETX across one-hop links to neighbors) and the CTP routing component (which computes the ETX across paths going through neighbors) maintain separate tables for their respective

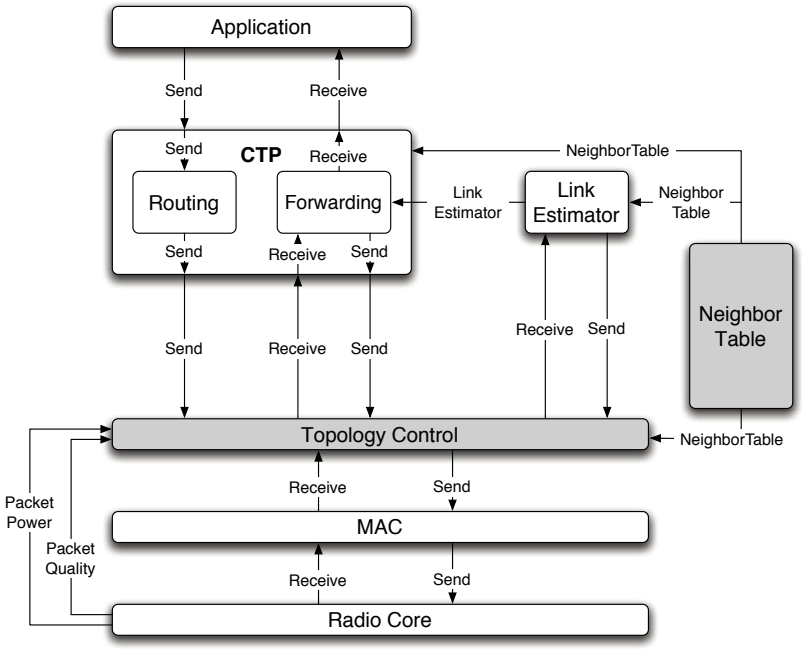


Figure 2.9: The augmented architecture; major new components are shaded

link quality data. This design choice increases the complexity of both components, since they must each include code to manage their own neighbor tables and to keep the two tables coherent. It also forces the `LinkEstimator` interface to include additional commands and events for the sole purpose of keeping the two tables coherent. As a result, although CTP and the beaconing link estimator are nominally independent components, as currently implemented they are tightly coupled.

We determined that extending this approach to include a third neighbor table (for the topology control layer) would be too clumsy. Instead, we extracted the neighbor table management code from CTP and the link estimator and used it to create a separate `NeighborTableC` component. Each entry in the table is now split into three "columns": one each for the link estimator, routing engine, and topology control. To flexibly support different link estimator, routing, and topology control components, each component defines a nesC `struct` type representing its own data (`link_estimator_data_t`, etc.) which the neighbor table treats as a black box.

We extracted all of the neighbor table management functionality from the `LinkEstimator` interface and moved it into a new `NeighborTable` interface, which simplified wiring and provided a better separation-of-concerns. The resulting architecture is shown in Figure 2.9; the link estimator, routing components, and topology control layer are now decoupled. The only significant inter-component dependency is

that CTP's forwarding engine uses the simplified `LinkEstimator` interface to query the link estimator component.

As discussed in Section 2.3.4, ART should send CTP's broadcast beacons at maximum power. Because TinyOS does not differentiate between multicast and broadcast traffic at the radio layer, the ART implementation instead approximates the desired behavior by treating CTP control packets as a special case and transmitting them at maximum power. While this approximation is specific to CTP, note that it does not introduce a compile-time dependency between CTP and ART: ART simply looks for a well-known constant in the TinyOS packet header which represents CTP control traffic.

Using this architecture, we implemented PCBL and ART as self-contained, platform-independent topology control layers. We also implemented a default topology control layer which simply passes through all packets untouched. Because these layers are self-contained, it is possible to interchange them at compile time using a compiler switch.

2.5 Experimental Results

In this section, we present an empirical evaluation of ART on our testbed of TelosB motes. We first measure the ROM and RAM overhead of our implementation of ART within TinyOS. We then evaluate ART's performance at the link level, and then compare ART's performance against PCBL in a data collection scenario[4]. Finally, we evaluate the effectiveness of ART's optimization for handling contention under heavy load.

Throughout this section, ART is deployed with a target PRR of 95% and a window size of 50 packets. Where not otherwise specified, the implementation of ART includes the contention-handling optimization described in Section 2.3.3. We use a neighbor table size of 32 entries in all experiments. Note that `NeighborTableC` includes code to evict old neighbor table entries, which was extracted from TinyOS's link estimator. Therefore in practice, the CTP default of 10 neighbors should be sufficient for most applications; we increased the table size to 32 rows for the purposes of this benchmark

[4]We did not include ATPC in this performance comparison, because the codebase used in [53] is not publicly available as of this writing, and ATPC's relative complexity made it impractical to reimplement.

	ROM	RAM
Max Power	17794	4614
ART	19376	5006

Table 2.1: The RAM and ROM overhead (bytes) of ART

in order to isolate the topology control layers' behavior from that of the eviction routine.

2.5.1 Memory Footprint

A primary goal of ART is to provide a robust topology control algorithm which can realistically be deployed on hardware-constrained sensor hardware. It is therefore important that ART can be implemented with realistically-low overhead on RAM and ROM consumption.

Table 2.1 compares the ROM and RAM usage statistics for the benchmark application described in Section 2.5.2 when compiled for the TelosB motes, with and without ART; these statistics are generated by the TinyOS toolchain. There is a 392-byte difference in RAM consumption between ART and the default (max power) topology layer. 384 of these bytes can be attributed to the 12-byte topology control data column stored in the 32-row neighbor table. As noted above, most applications will not need a neighbor table of this size and will see a proportionally smaller memory overhead. The ROM overhead is larger at 1582 bytes, which is insignificant when compared to the ROM size of representative sensor hardware (e.g., 48 KB for TelosB).

2.5.2 Link-Level Performance

To examine the impact of ART on a per-link basis, we performed the following benchmark. 29 links are selected at random from the 524 links detected in the testbed during Section 2.2.2. A benchmark application cycled through these links round-robin, sending 100 packets over the one-hop link each time it was selected. Since the benchmark transmits only over a single link at a time, there is minimal contention. This cycle repeated for 150 rounds over the course of 24 hours. We performed this benchmark with no topology control (i.e., maximum power) and with the ART topology control layer; both benchmark runs used the same 29 links.

	PRR	Avg. Current
Max Power	56.7% ($\sigma = 2.5\%$)	17.4 mA ($\sigma = 0$)
ART	58.3% ($\sigma = 2.1\%$)	14.9 mA ($\sigma = 0.32$)

Table 2.2: The link-level performance of max-power and ART

Table 2.2 presents the overall results. Max-power and ART have an insignificant difference in PRR results, demonstrating that ART indeed selects power levels equivalent in PRR to the maximum power setting. (Because there is minimal contention in this benchmark, ART cannot achieve a significant increase in PRR against max-power.) ART achieves this with a 15% average reduction in current consumption over max-power.

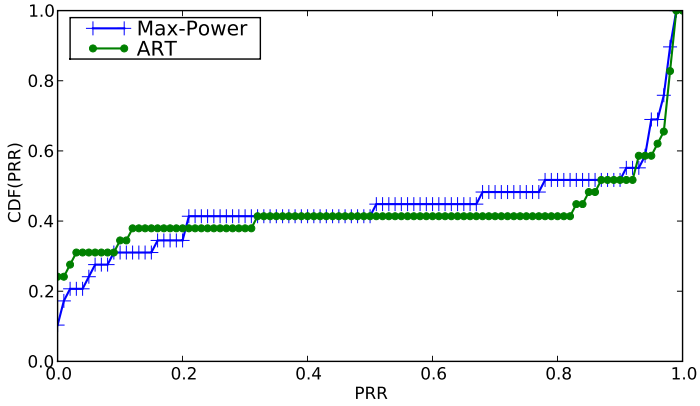


Figure 2.10: The PRR distribution under max-power and ART

ART's overall PRR of 56.7% is significantly lower than this target PRR of 95%. This occurs because, even at maximum power, there is a bimodal distribution of link qualities (shown in Figure 2.10). For example, 15 of the 29 links in this experiment achieve a PRR $\geq 90\%$, while 9 of the 29 links achieve a PRR $\leq 10\%$. Note that ART and max-power have similar PRR distributions, again indicating that ART achieves similar PRR to max-power even on links where it is unable to meet its target.

We now take a closer look at the ART's behavior over one interesting link in the testbed, which is generally high-quality but still shows some link quality fluctuation. The PRR and average current consumption of this link are shown in Figure 2.11. We see that ART is able to lower the link's current consumption by an average of 2.3 mA, responding to link quality fluctuations by tuning the power level accordingly. Of particular interest is ART's behavior during round 10 and rounds 120–140, when link quality sharply drops and ART attempts to salvage the link by quickly going to maximum power. As a result, ART achieves an overall PRR of 93.7% across this

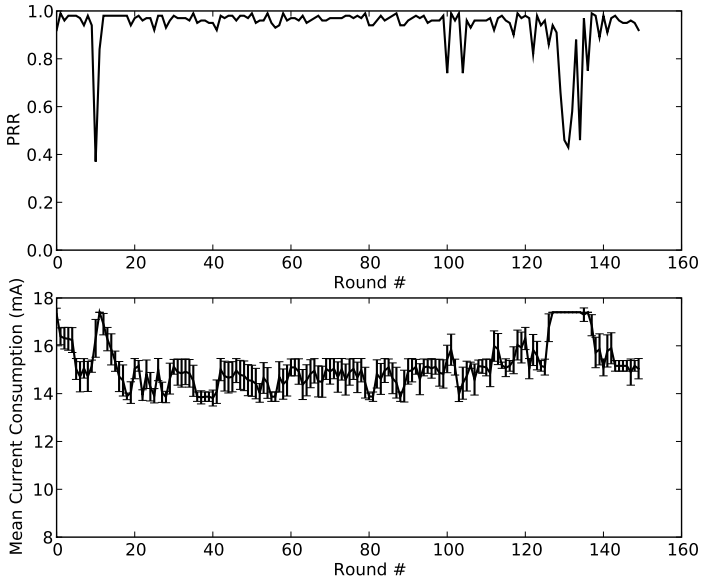


Figure 2.11: The behavior of link 129 → 106 under ART

link, close to the target PRR of 95%. ART performs slightly below the target overall because of these two temporary but sharp drops in link quality.

2.5.3 Data Collection

We evaluated the performance of ART against max-power PCBL on a multi-hop data collection application built on top of the CTP [34] routing library. To get a better understanding of the link-layer packet loss, we disabled CTP's automatic packet retransmission routine. The application designated a particular node in the testbed as the tree's root, and then waited 5 minutes for the routing layer to bootstrap. It then selected one node from the testbed and instructed it to send 200 data packets to the sink node, which recorded the sequence number and hop count of all packets it received. After the sender was finished with its 200 packets, another sender node was selected in a round robin fashion. We performed this experiment for 9 rounds over 4 hours at max-power, and then repeated the experiment with PCBL and ART.

We reimplemented PCBL using the architecture described in Section 4.3. We configured PCBL to use the thresholds of 90% and 98% identified in [72]; similar thresholds were found in our own testbed (see Section 2.2.4). To simplify PCBL's implementation, we performed its bootstrapping procedure offline using 200 packets per node per power level.

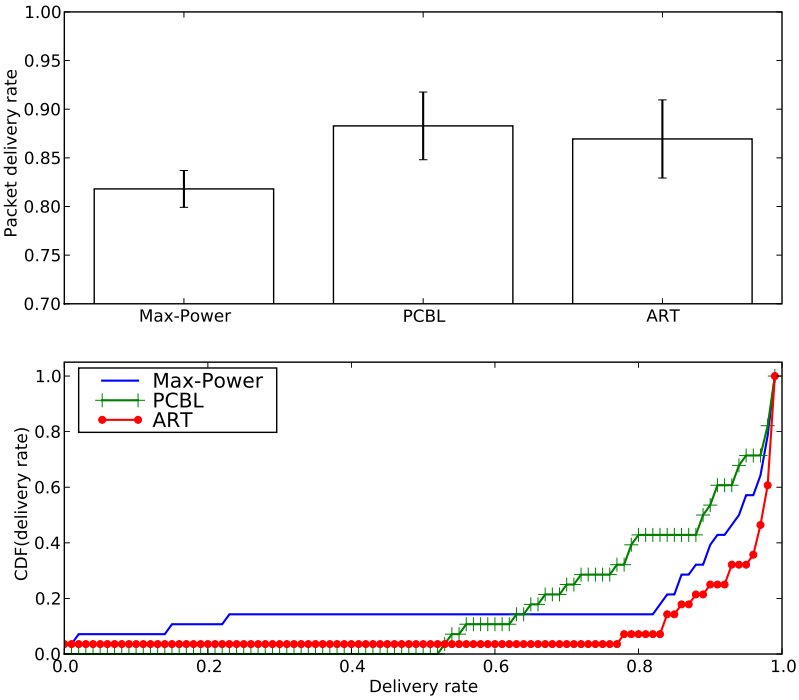


Figure 2.12: The end-to-end delivery rate of max-power, PCBL, and ART under low contention

Figure 2.12 shows the end-to-end delivery rate under these three schemes. Both ART and PCBL achieve good PRR in this experiment, outperforming the max-power scheme by 6.4% and 5.1%, respectively. PCBL collects a large amount of link quality data up-front, allowing it to blacklist poor-quality links and prevent CTP from ever considering them. ART achieves comparable performance to by reducing transmission power, which reduces intra-path contention even when there is only one node sending at a time. It is worth emphasizing that ART achieves this PRR without the need for PCBL's extensive bootstrapping phase. Also note that 75% of the sources achieve a delivery rate of 90% or higher under ART, compared to 61% under max-power and 46% under PCBL.

Looking closely at the distribution of PRRs among the senders, max-power has starved three of the senders with the highest average hop-counts (see Figure 2.13). This occurs because, although there is only one node producing data at a time, CTP will allow the application to produce a new packet as soon as the previous packet is one hop away from the sender. Therefore, a single sender may contend with its own packets which are still traversing a multi-hop path to the sink. This self-contention

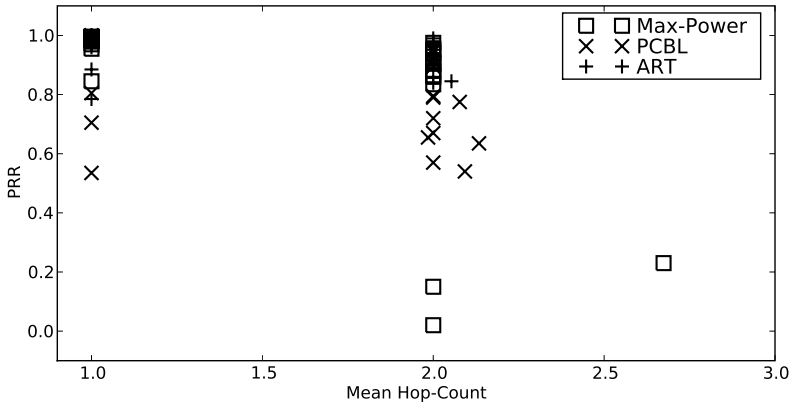


Figure 2.13: The relationship between PRR and hop count under max-power, PCBL, and ART

effect is the most pronounced when all packets are sent as maximum power, resulting in starved nodes. This finding underscores the importance of transmission power control, especially in multi-hop networks.

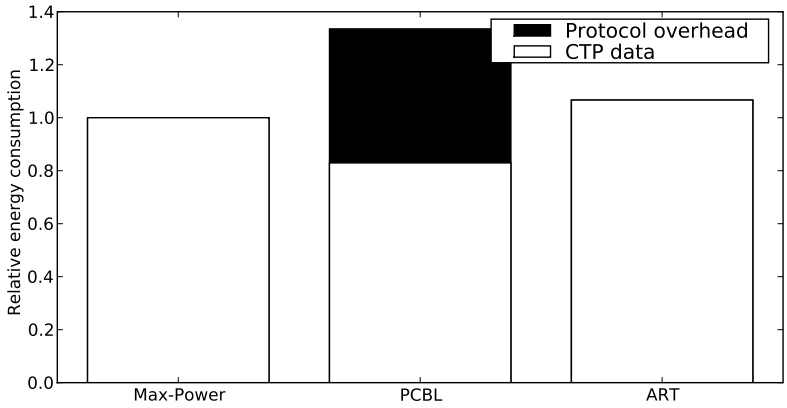


Figure 2.14: The energy consumption of max-power, PCBL, and ART under low contention

Figure 2.14 illustrates the total energy consumed by packet transmissions during each of these benchmark runs, normalized to the max-power energy consumption. ART has an energy consumption 6.6% higher than that of the max-power topology. This increase in power consumption occurs because for two reasons. First, ART has a 6.4% higher PRR than max-power and therefore transmits proportionally more packets through intermediate nodes. Second, as shown in Figure 2.13, max-power has starved the three nodes with the most expensive paths to the sink, which decreases total energy consumption at the expense of these senders.

Excluding its bootstrapping cost, PCBL achieves the lowest energy consumption, with a reduction of 17% compared to max-power. PCBL's bootstrapping cost constitutes a 60% energy overhead in this benchmark; the relative overhead will decrease the longer the application remains active without rebooting PCBL. We project that PCBL would have achieved equal energy consumption to ART if the benchmark were extended to 8 hours and link conditions remained stable. Note also that rebooting PCBL can disrupt the network for extended periods of time: the bootstrapping phase took over 2 hours to complete in our testbed.

2.5.4 Handling High Contention

To explore the impact of ART's contention handling optimization, we performed an experiment similar to that in Section 2.2.3. Ten links selected at random from the testbed simultaneously sent data in batches of 200 packets; this procedure was repeated for 30 minutes. (The same set of ten links was used throughout all benchmark runs; in the interest of fairness to PCBL, we verified that none of the ten links had been blacklisted.) We performed this experiment under the max-power, PCBL, and ART topology control schemes. In order to isolate the effect of ART's contention-handling "gradient" optimization, we also repeated the benchmark with this optimization disabled.

Since we also wished to capture the effect of dynamic workload changes on PCBL's behavior, we reused the PCBL bootstrapping data collected for the previous experiment. Accordingly, we do not include PCBL's bootstrapping overhead when calculating energy efficiency.

Figure 2.15 shows the PRR of these benchmark runs. The difference in PRR between max-power (83.6%) and the unoptimized ART (83.9%) is insignificant. This occurs because the packet loss is too high for the unoptimized ART to ever leave the maximum power setting, and so its behavior is essentially identical to that of max-power. As shown in Figure 2.16, the unoptimized ART achieves only 5.1% energy savings over max-power for similar reasons.

The optimized ART achieves lower PRR (66.1%) than max-power, indicating that it cannot locate the optimal transmission power. This happens because there are many nodes which are rapidly sending packets and dynamically adjusting their transmission

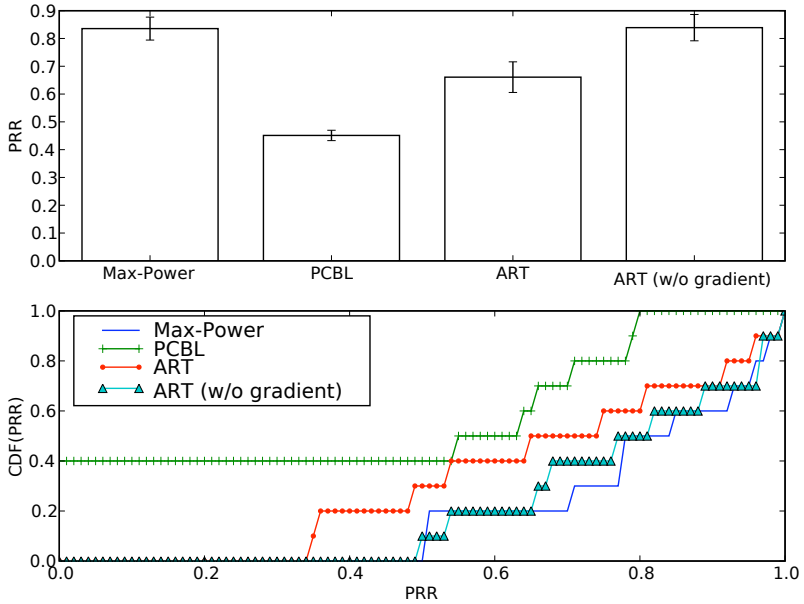


Figure 2.15: The PRR of max-power, PCBL, and ART under high contention

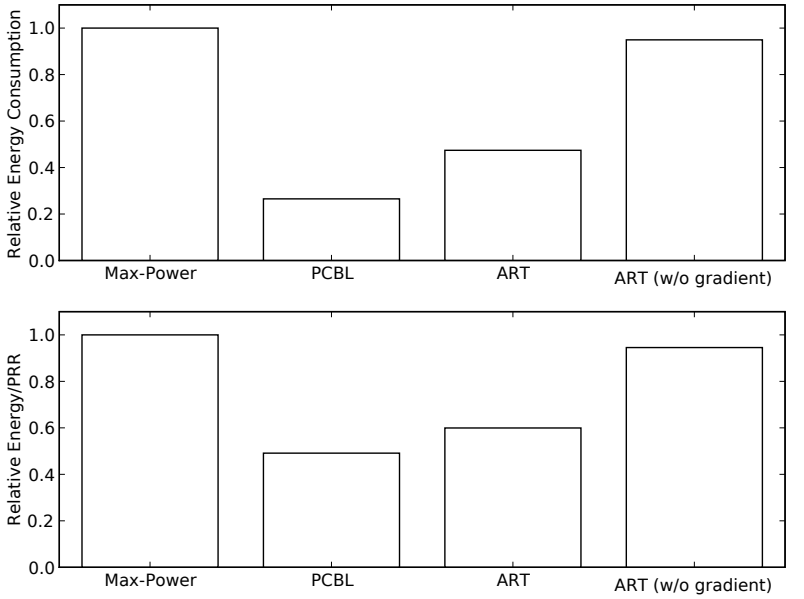


Figure 2.16: The energy consumption and efficiency of max-power, PCBL, and ART under high contention

powers, both of which have a significant effect on the effective link quality. ART's sliding window mechanism cannot effectively track such rapid link quality fluctuations. Nevertheless, as shown in Figure 2.16, the optimized ART consumes only 47.4% that of max-power's energy. As a result, ART's energy efficiency (i.e., the average cost of successfully transmitting one packet) is 40.0% better than max-power.

PCBL achieves the lowest PRR (45.1%) and lowest energy consumption (26.5% that of max-power) among all four schemes. While this makes PCBL 50.9% more energy efficient than max-power and 18% more energy efficient than the optimized ART, it does so at the expense of starving four of the ten links. These four links had very good link quality during the bootstrapping phase, and so PCBL assigned them low transmission powers (two were assigned the lowest possible setting, while the other two were assigned the third-lowest setting). Under high-contention workloads, receiver nodes will overhear transmissions from nearby high-power transmitters and be unable to receive packets from these low-power transmitters.

Chapter 3

Decentralized Structural Damage Localization

The deterioration of our civil infrastructure is a growing problem both in the US and around the world. For example, during their lifetimes, bridges suffer from environmental corrosion, persistent traffic and wind loading, extreme earthquake events, material aging, etc., which inevitably result in structural deficiencies. According to the American Society for Civil Engineers 2009 Report Card for America's Infrastructure, "more than 26%, or one in four, of the nation's bridges are either structurally deficient or functionally obsolete" [5]. Due to the expense of retrofitting a structure with a wired sensor infrastructure, most of these structures are not currently being continuously monitored.

Structural Health Monitoring (SHM) aims to determine the condition of a civil structure, provide spatial and quantitative information regarding structural damage, or predict the performance of the structure during its lifecycle. Recent years have seen growing interest in SHM based on wireless sensor networks (WSNs) due to their potential to monitor a structure at unprecedented temporal and spatial granularity. WSNs permit a dense deployment of measurement points on an existing structure, facilitating accurate and fault-tolerant damage identification techniques without the need to install a fixed wired infrastructure [62]. Indeed, numerous SHM systems have been proposed in literature which leverage WSNs to collect raw sensor data [16, 18, 42, 90]. These systems are generally designed to support traditional centralized SHM methods, with special consideration to the limited bandwidth and energy supplies that are not present under a traditional system of wired sensors.

However, by treating SHMs as a simple data collection devices for supporting centralized SHM methods, the resulting systems inherently suffer from high energy consumption and prolonged detection latencies. For example, a state-of-art system deployed at the Golden Gate Bridge required 9 hours to collect a single round of data from 64 sensors, resulting in a system lifetime of 10 weeks when using four 6V lantern batteries as a power source [64]. This system's high latency and relatively short lifetime arose from the fact that the underlying SHM method was designed separately from the WSN system. Specifically, the SHM method required the WSN to reliably deliver the entire raw sensor dataset to the base station for centralized processing, inherently placing a high network burden on the WSN system.

What is needed is a fundamentally different *co-design* approach which considers both the constraints of the underlying WSN system and the SHM requirements in its numerical approach. This can be achieved by leveraging the increasingly powerful processing capability of wireless sensor "motes" to partially process locally-collected data, extracting (and subsequently exchanging) only the important features relevant for SHM. Several recent studies demonstrate the potential for distributed SHM approaches to significantly reduce energy cost through localized data processing [15, 62, 99].

In this chapter, I discuss my work on a decentralized SHM system based on the *Damage Localization Assurance Criterion* (DLAC) algorithm [59, 60]. In contrast to centralized approaches that require transporting large amounts of sensor data to a base station, the system discussed here pushes the execution of portions of the damage localization algorithm onto each sensor. This in-situ processing results in significant reductions in communication overhead and energy consumption, while consuming only a small fraction of resources available on the off-the-shelf Intel Imote2 [26] sensor platform. This chapter discusses the following specific contributions:

1. a damage localization system that integrates a decentralized computing architecture optimized for the DLAC algorithm;

2. a proof-of-concept implementation of the DLAC-based design using the TinyOS operating system [1]; and

3. empirical results and analysis that demonstrate that DLAC can accurately detect and localize damage on a simple beam structure and on a complex truss

structure, while significantly outperforming a centralized approach in terms of latency, energy efficiency, and system lifetime.

This DLAC-based system represents the first step toward a *co-design* approach to cyber-physical system (CPS) design for SHM. In Chapter 4, I discuss a further evolution of this co-design approach, based on a more powerful numerical approach supported by a hierarchical network architecture.

3.1 Related Work

A UC Berkeley project to monitor the Golden Gate Bridge [42] represents one of the first large-scale deployments of smart sensor networks for SHM purposes. Vibration data is collected and aggregated at a base station under a centralized network architecture, where frequency domain analysis is used to perform modal content extraction. However, it took nearly a full day to transmit sufficient data for such computations. Similarly, researchers at Clarkson University have implemented a wireless sensor system for modal identification of a full-scale bridge structure in New York [31]. Battery-powered wireless sensor nodes equipped with accelerometers and strain transducers are used, having a high wireless data transmission rate. The entire network is polled by a master computer that collects acceleration and strain data. Both modal identification and quantification of static responses are performed using a centralized network architecture. Wisden [90] provides services for reliable multi-hop transmission of raw sensor data, using run-length encoding to compress the data before transmission. These centralized approaches suffer from two fundamental limitations. First, data may only be collected from a limited number of nodes in a reasonable time frame, which would allow the system to only detect the most severe (and probably visually apparent) damages. Second, such systems are inadequate for timely detection of structural failures resulting from extreme events (e.g., earthquakes) due to the prolonged time needed for collecting and analyzing data.

BriMon [18] partially addresses the communication bottleneck by sampling data at 400 Hz and averaging this data over 40 Hz windows. The data resolution and network size (a maximum of 12 nodes per span) supported by BriMon may not be fine-grained enough for damage detection and localization on complex structures. A deployment

in the Torre Aquila heritage building [16] uses lossless compression to deliver heterogeneous sensor data to sink node. The network burden of this deployment was eased by the specific kinds of data needed to monitor the building's health: only three acceleration sensors were required, while the environmental and deformation sensors produced only 1–10 readings every 10 minutes.

The above limitations motivate the need for a *co-design* approach which addresses both the SHM and WSN concerns in a holistic manner. An integral part of such a solution is the adoption of distributed SHM solutions [55, 75]. Researchers at the University of Illinois at Urbana-Champaign have experimentally validated a SHM system that employs a smart sensor network deployed on a scale three-dimensional truss model [62,74]. Results demonstrate that the adopted SHM system is effective for damage identification and localization; however, significant communication is involved in performing data cross-correlation, which results in significant energy consumption.

[84] implemented a low-cost and rapid-to-deploy wireless structural monitoring system on a long-span cable-stayed bridge in Taiwan. The full-scale test was conducted by collecting ambient vibration data of the bridge and analyzing it in situ by two modal identification methodologies, the stochastic subspace identification method (SSI) and frequency domain decomposition method (FDD). Modal ID results led to the determination of a total of 10 modal frequencies and corresponding mode shapes within a frequency range of 0–7 Hz. [94] also implemented an automated modal identification by optimizing output-only modal methods (FDD with peak-picking) for a distributed wireless sensor network. The distributed implementation, tested in a balcony of a theater, used a parallel data processing and reduced communication scheme to ensure scalability and power efficiency in the WSN. In their implementation, three network topologies are proposed to yield a two-node based data sharing chain. This implies the partial mode shape identified from each pair of nodes has to be recombined to recreate the complete mode shape necessary for damage detection. However, this strategy would potentially amplify the recombination error, if any one of the sensor nodes is unreliable.

3.2 Design and Implementation

In this section, we describe the design of an SHM system based on the DLAC algorithm. We first summarize the DLAC algorithm, focusing on features that make

it a compelling candidate for decentralized processing on wireless sensors. We then describe a decentralized architecture specifically optimized for this damage localization algorithm. A salient feature of this architecture is the partitioning of the damage localization algorithm between the wireless sensors and the base station, which significantly reduces the sensors' communication load and energy consumption in exchange for moderate processing costs on each sensor. Finally, we discuss an implementation of this architecture on top of the TinyOS [1] operating system.

3.2.1 Damage Localization Algorithm

The decentralized system described in this chapter is based on the *Damage Localization Assurance Criterion* (DLAC) technique [59, 60], which analyzes data collected at each sensor to detect and localize structural damage. This subsection provides an overview of the DLAC algorithm, focusing on the details that motivate its use for a codesigned SHM system. The DLAC algorithm itself is existing work from the structural engineering field, and is not part of this dissertation's contribution.

The DLAC algorithm is especially well-suited for a decentralized WSN system [15,22], because it performs damage localization based on post-processed natural frequency data rather than raw vibration data. As discussed below, this natural frequency data is computed from each node's raw vibration data (i.e., accelerometer readings). Moreover, nodes do not need to correlate individual sensor readings to compute this natural frequency data, which would require precise time synchronization across nodes.

The damage localization process includes an offline phase and an online phase. In the offline phase, the system identifies the natural frequencies of the healthy structure, using observed vibration (acceleration) data and a series of transformations described below. Because these natural frequencies change in response to structural damage, they are an effective "signature" of the structure's health. (The natural frequencies are uniform throughout the entire structure, and so even localized damage will produce a global change in the frequency content.) Additionally, as required by the DLAC technique, an analytical model of the structure and the estimation of its natural frequencies using purely numerical techniques are performed. By comparing the observed natural frequencies against those estimated by the numerical model, we are effectively able to capture the numerical errors generated by the imperfect model.

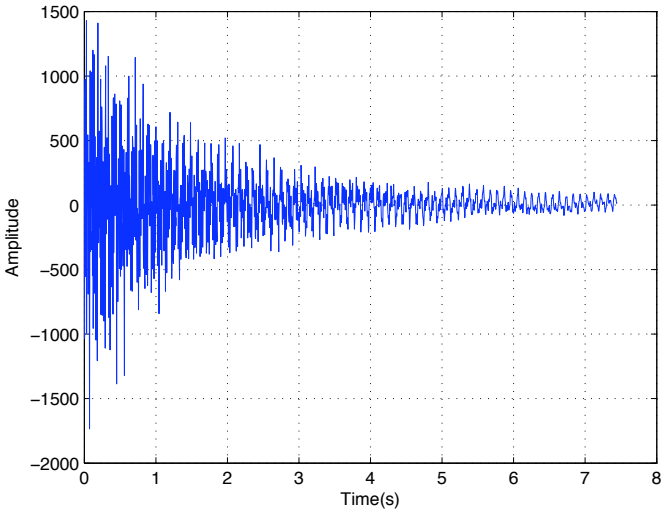


Figure 3.1: Raw vibration readings taken after exciting a steel beam with a hammer

In the system's online phase, it periodically samples new vibration data. An example of a raw sensor reading, taken during the experiment described in Section 3.3.1, is shown in Figure 3.1. The natural frequency identification procedure is repeated on this newly-collected data. In the final stage of the algorithm, this new frequency data and the structure's analytical model enable the DLAC algorithm to localize the damage to discrete locations on the structure.

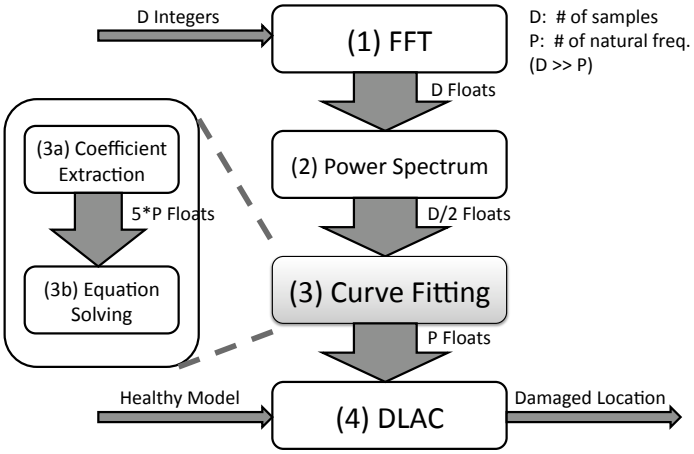


Figure 3.2: The online phase of damage localization

The online phase of our system can be decomposed into four stages, which are summarized in Figure 3.2. Steps (1)-(3) are used to compute the current natural frequencies of the structure based on collected vibration data, which are then input into the DLAC algorithm in Step (4).

(1) The raw sensor readings are converted from time domain data to frequency domain data using a **Fast Fourier Transform** (FFT). This produces a series of complex numbers as output, represented as an array of floating point numbers twice the length of the original input (one real and one imaginary part per input). A property of the FFT output data is that its magnitudes are symmetric. To save memory and computation in later stages, the mote discards the redundant half of this frequency domain data, resulting in a final output the same length as the input.

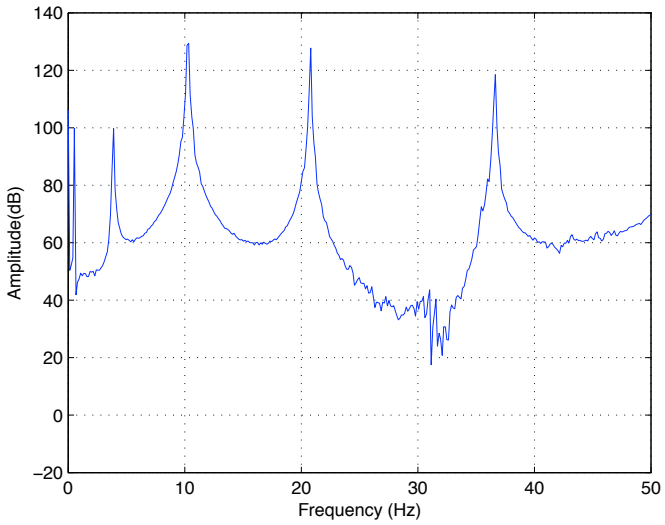


Figure 3.3: Power spectrum analysis results of raw vibration data, with the redundant upper half already removed

(2) The FFT's output is fed into a **power spectrum analysis** routine, which calculates the squared magnitude of each complex value in the FFT output data. Figure 3.3 demonstrates the output of power spectrum analysis over the previous raw sensor data trace.

(3) The natural frequencies in this power spectrum data are identified by performing **polynomial curve fitting**. The goal of this process is to identify the frequency values associated with the peaks in the power spectrum curve for each mode. Empirical study has shown that the Fractional Polynomial Curve-Fitting (FPCF) technique is reliable for identifying a structure's modal frequencies in an automated manner. FPCF fits the power spectrum data to a polynomial function in the form of Equation 3.1, with the order of its denominator proportional to the number of frequencies we wish to locate. This function was identified in [50] to extract features from system transfer functions, and represents both a smoothing and an interpolation of the raw power spectrum data.

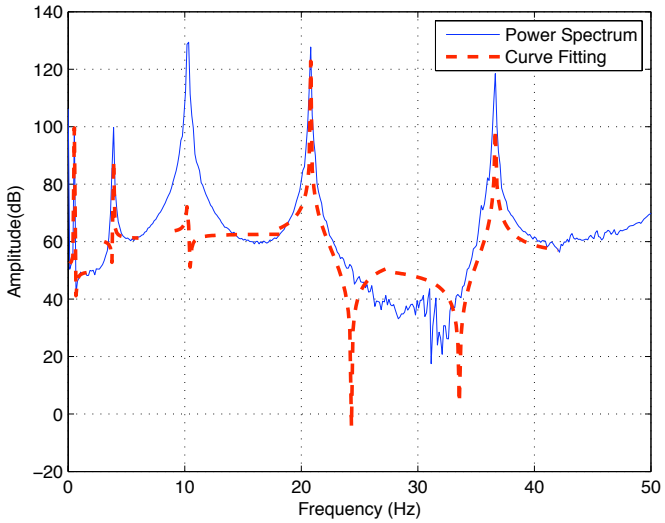


Figure 3.4: Polynomial curve fit to the power spectrum analysis data

$$H(s) = \frac{B(s)}{A(s)} = \frac{b_1 s^m + b_2 s^{m-1} + \ldots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + \ldots + a_{n+1}} \tag{3.1}$$

Figure 3.4 illustrates the results of fitting a 2nd-order curve near each separate peak in the power spectrum data discussed above. Note that, as in Figure 3.4, the fitted curve does not necessarily match the amplitude (Y-axis) of the power spectrum data at all of the peaks. The goal of this step is to obtain the imaginary parts of the roots of Equation 3.1's denominator, which correspond to the frequencies of the structure (X-axis); the amplitude of the fit is therefore irrelevant.

For the purposes of implementation and analysis, the identification of natural frequencies may be subdivided into two steps: (3a) **coefficient extraction**, which represents the curve-fitting problem as a series of matrices; and (3b) **equation solving**, which applies the matrix operations necessary to determine the roots of the denominator polynomial.

(4) Finally, once the structure's natural frequencies have been identified, they are used as input into the **DLAC** algorithm, which ultimately detects and localizes damage to the structure. The DLAC algorithm also uses the structure's numerical model to simulate damage at discrete locations along the structure, providing an estimate of how the natural frequencies *would* change in response to damage at each of these locations. Finally, DLAC uses the structure's healthy frequency data (both the observed and predicted values) to capture and accommodate errors in the numerical

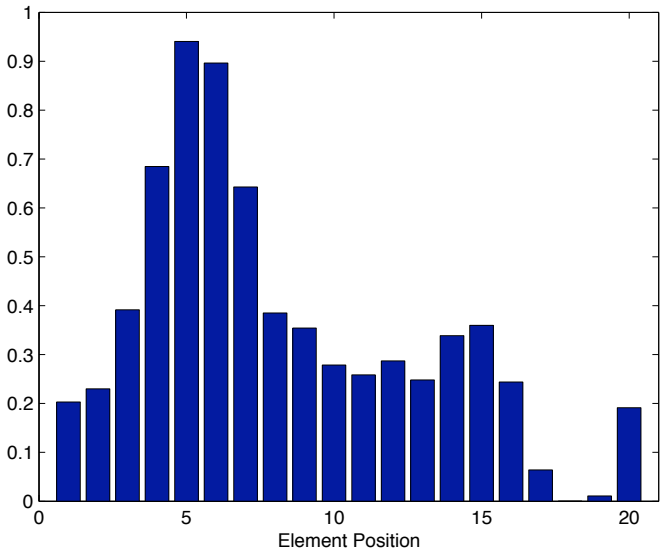


Figure 3.5: DLAC results representing the correlation of damage to 20 discrete locations along a steel beam; higher numbers represent a greater likelihood of damage

model. Based on these inputs, DLAC yields a vector of numbers in the range $[0, 1]$, representing the correlation factors to damage at various discrete locations along the structure. In Figure 3.5, we plot $DLAC$ for a steel beam that has been subdivided into 20 discrete regions; relatively high DLAC values concentrated around $X = 5$ indicate a strong correlation with damage at the fifth region.

3.2.2 Decentralized Architecture

We have developed a decentralized computing architecture specifically optimized for the damage localization procedure presented in Section 3.2.1. Our structural health monitoring system consists of low-power sensor motes and a base station connected by a wireless network. Due to the difficulty of replacing batteries for sensors embedded in a structure, the sensors' energy efficiency is a critical concern for SHM systems. In contrast, the base station (typically a PC) is connected to a wired power source and has significantly more resources than the sensors. Each mote collects raw vibration data from an attached accelerometer and performs parts of the damage localization procedure. The motes transmit their partial results wirelessly to the base station, which completes the damage localization procedure.

With the advance of sensor hardware, commercial sensor platforms such as the Imote2 are capable of moderate amounts of in-network processing. Our decentralized architecture exploits these processing capabilities to reduce the communication and energy costs of damage localization. Because portions of the damage localization procedure described in Section 3.2.1 (e.g., the DLAC algorithm) involve complicated optimization routines, it is impractical to perform damage localization entirely on the motes. However, offloading too much computation onto the base station would require transmitting large amounts of data, on the order of thousands of floating-point numbers. An important design goal of our system was therefore to find the proper balance between the time and energy spent on computations on the motes, and the time and energy spent sending partial results to the base station.

To identify the optimal partitioning between the motes and the base station, we analyze here the data flow between stages of the damage localization procedure. We validate our analysis through a comprehensive empirical measurement of different partitioning strategies in Section 3.4. As shown in Figure 3.2, we parameterize this analysis by the number of samples being collected, D, and the number of frequencies to identify, P ($D \gg P$). The FFT stage consumes D integer sensor readings as input, and produces D floating-point values as output. Power spectrum analysis transforms these D floating-point values into $\frac{D}{2}$ floating-point magnitudes. The coefficient extraction portion of the curve-fitting routine represents the power spectrum data as $5P$ floating-point coefficients; applying the equation solver reduces this to P floating-point values.

As shown by the detailed empirical evaluation in Section 3.4, partitioning between the curve fitting and DLAC stages results in an optimal energy efficiency and latency. The curve fitting routine results in significant reduction in the amount of data that must be transferred to the next stages, from the hundreds or thousands of collected vibration samples to a single vector of size P. For a typical setup of $D = 2048$, $P = 5$, 16-bit accelerometer readings, and single precision (32-bit) `float` types, the stages before curve fitting generate from 4 KB to 16 KB of data; in comparison, curve fitting outputs only 20 B. In practice, the relatively complex equation solving substage of the curve fitting routine may be impractical to implement on some sensor network platforms. The system may alternatively be partitioned between the coefficient extraction and equation solving substages of the curve fitting routine, which outputs $5P$ matrix coefficients (100 B of data under the setup described above). Based on our detailed empirical analysis described in Section 3.4, the in-situ processing performed before

either partitioning point reduces the communication latency so that the raw data collection stage dominates the algorithm's running time. Similarly, the radio's energy consumption is then dwarfed by the cost of idle sleeping when either partitioning point is selected, and represents 0.98% or less of the system's total energy budget. This partitioning of the damage localization procedure between the motes and the central base station highlights the importance of an integrated design for the computing architecture and the damage localization techniques.

3.2.3 Implementation

This architecture is implemented as a proof-of-concept SHM system containing two major software packages, which are available as open-source software at [2]. The first package is implemented on top of the TinyOS 1.1 operating system, and is deployed on the Imote2 hardware platform. The Imote2 motes are equipped with 32 MB of RAM, XScale CPUs capable of running at speeds up to 614 MHz, and add-on sensor boards with integrated accelerometers [25].

The current implementation assumes that sensors are within a single hop from the base station, as the focus of this work is on decentralized processing rather than network protocols. However, it can easily be extended to support multi-hop networks by incorporating existing multi-hop data collection protocols [34, 42]. Section 3.4.4 discusses the implications of multi-hop networking on system lifetime.

The second software package consists of a Java application and MATLAB scripts running on the base station PC. A GUI allows users to set the algorithm's parameters, initiate data collection and aggregation on individual motes, and collect the partial curve fitting results computed by the motes. Once the application receives partial results from a mote, it completes the curve fitting procedure using an equation solver written in Java. The results of this equation solver are then processed using a MATLAB script that implements the DLAC algorithm. For debugging purposes, the last set of raw sensor readings may be retrieved from individual motes; this feature is not used under normal operations.

To simplify the implementation, the SHM algorithm is currently invoked only when requested by the PC-side GUI. The motes currently keep their radio on to listen for these control messages, which can rapidly deplete their batteries. We emphasize that

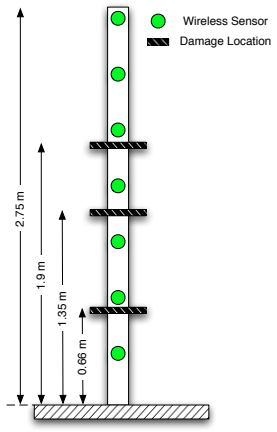


Figure 3.6: Diagram of cantilever beam test structure

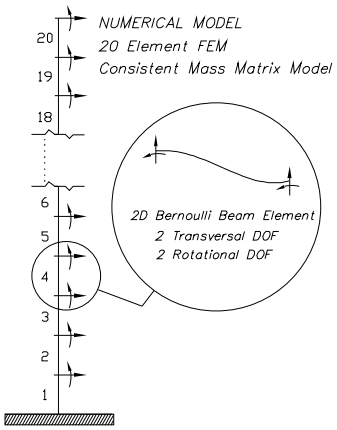


Figure 3.7: Cantilever beam finite element model

there is nothing *inherent* in our decentralized approach that prohibits performing autonomous readings at prescheduled intervals and/or managing the radio power, e.g., by using existing power-efficient MAC protocols. Section 3.4.4 discusses these options in greater detail.

3.3 Evaluation: Damage Localization

In this section, we present an evaluation of our SHM system's *numerical* performance, discussing our system's ability to localize damage on two sample structures. The two structures' different physical properties serve as good indicators of DLAC's performance under ideal and complex conditions, respectively.

3.3.1 Beam

To validate our damage localization system, we first performed a series of experiments on a steel cantilever beam in the Structural Control and Earthquake Engineering Lab at Washington University in St. Louis. The beam, depicted in Figure 3.6, is 2.75 m long, 7.6 cm wide, and 0.6 cm thick and fixed to the ground to approximate a cantilever support. Damage along the beam can be simulated at three distances from the beam support by attaching a 1.5 kg steel bar. Because this beam has relatively simple structural properties, it serves as a test of our system under ideal conditions.

Mode	**1**	**2**	**3**	**4**	**5**
Measured	0.5381	4.0240	11.4705	22.5506	37.4316
Analytical	0.6564	4.1133	11.5180	22.5710	37.3160

Table 3.1: Measured and analytical natural frequencies for the healthy beam

Mode	**1**	**2**	**3**	**4**	**5**
Analytical	0.6555	4.0105	10.6192	20.8768	36.1469
Sensor 1	0.5506	3.9043	10.2473	20.7641	36.6415
Sensor 2	0.5374	3.8902	10.2779	20.8069	36.6396
Sensor 3	0.5402	3.8977	10.2714	20.7964	36.6048
Sensor 4	0.5316	3.8564	10.2744	20.8470	36.6785
Sensor 5	0.5371	3.7678	10.0707	20.4038	36.9797
Sensor 6	0.5427	3.8488	10.3217	20.7546	36.5919
Sensor 7	0.5392	3.9012	10.2533	20.7751	36.6570

Table 3.2: Analytical and identified natural frequencies for the damaged beam

We collected data about the beam's healthy state by attaching seven Imote2 wireless sensors at equidistant intervals along the beam. Each mote was equipped with a Crossbow ITS400 sensor board with embedded 3-axis accelerometers; tests on a shake table confirmed that these accelerometers are sufficiently accurate for DLAC purposes within their saturation range of $\pm 2.0g$. After exciting the beam with a hammer, we collected vibration data from each mote. Using this data, we determined the beam's healthy natural frequencies offline, as shown in Table 3.1.

A corresponding 2D Bernoulli beam model was generated in MATLAB, which subdivided the beam into 20 elements with 42 global degrees of freedom (Figure 3.7). As shown in Table 3.1, the first natural frequency predicted by the model is within 22% of the experimental value, while the other predicted frequencies fall within 2% of the experimental data. These discrepancies can be explained by simplifying assumptions in the model; e.g., the Imote2 nodes were not included in the model. We remind the reader that the DLAC algorithm uses both measured data and analytical data as inputs, thus accounting for such discrepancies.

We then tested our system's ability to detect and localize damage along the beam structure. Using the procedure described in Section 3.2, we collected and analyzed vibration data at 280 Hz, both in its healthy condition and with the steel bar attached at each of the three damage locations shown in Figure 3.6. We added an arbitrary amount of mass at each position in our analytical model to develop the matrix of damage cases for computation of the correlation factors. The amount of mass that

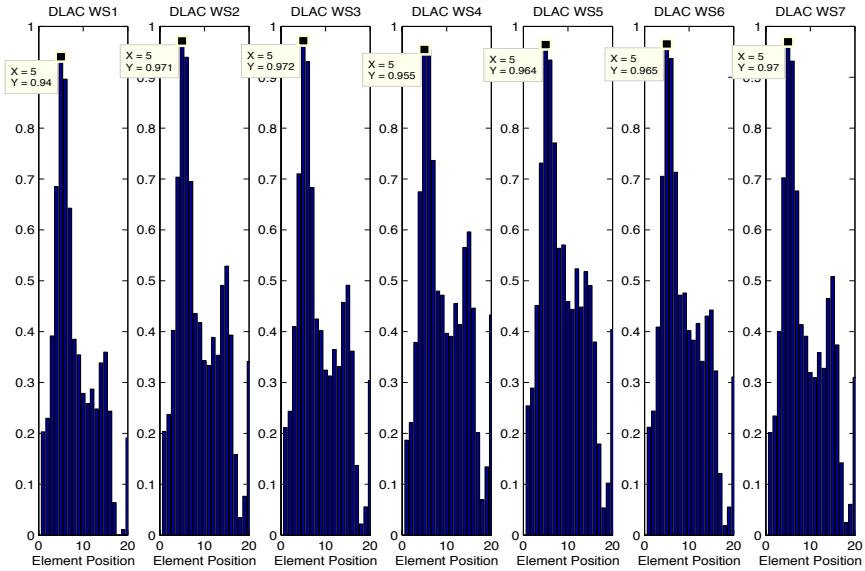


Figure 3.8: DLAC results for the beam damaged at element 5

we added to the model intentionally did not match the steel bar's actual mass. We included this discrepancy to reflect the fact that the amount of damage to a structure is not known ahead-of-time, and to illustrate that DLAC will still adequately localize damage as long as a reasonable guess is used.

For the sake of brevity, we present here only the results for the first scenario, which simulates damage at the beam's fifth element. As shown in Table 3.2, the natural frequencies measured by each of the 7 sensor nodes closely match those predicted by the "damaged" analytical model. Each node therefore correctly predicts structural damage at the beam's fifth element with a correlation of 94% or higher (Figure 3.8). We observed similar results during the other two damage scenarios, with the nodes consistently localizing the damage at the correct element with correlations of 90% or higher.

3.3.2 Truss

To evaluate our system under more complex structural configurations, we then performed tests on a 5.6 m steel truss structure [21] at the Smart Structure Technology Laboratory (SSTL) at the University of Illinois at Urbana-Champaign (see Figure 3.9). 11 Imote2 sensors were deployed on the frontal panel of the truss, as shown in Figure 3.10; USB cabling was deployed to power the motes, but all communication

Figure 3.9: 3D truss test structure

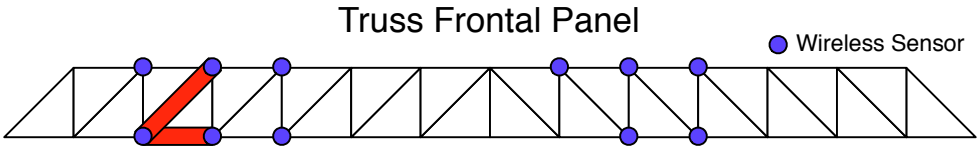


Figure 3.10: Truss experimental setup; highlighted elements were replaced to simulate damage

Mode	1	2	3	4	5
Measured	20.65	41.49	64.59	69.41	95.51
Analytical	19.88	38.31	66.26	67.17	92.25

Table 3.3: Measured and analytical natural frequencies for the healthy truss

occurred over their wireless radios. The truss consists of fourteen 0.4 m-long bays and sits on four rigid supports. Different structural configurations and damage scenarios can be emulated by removing or replacing the truss's members and its supports.

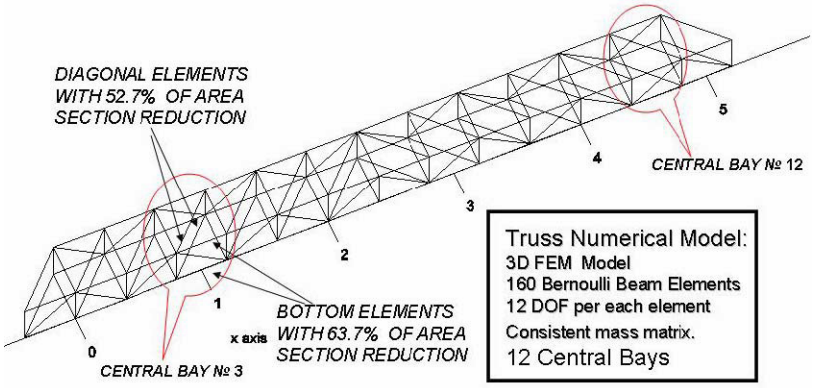


Figure 3.11: Truss finite element model

As with the beam, we used data collected from the healthy truss and a MATLAB model to compute the natural frequencies in the truss's healthy state. We collected the data by vertically exciting the truss structure using a magnetic shaker. (To ensure a consistent mass distribution with later experiments, the Imote2 motes were left installed but were not activated.) A force transducer was used to measure the input force, and six wired sensors were used to measure the vibrations at different points on the truss's frontal panel. A corresponding numerical finite element model with 160 beam elements and 336 global degrees of freedom (Figure 3.11) was generated in MATLAB. As shown in Table 3.3, the natural frequencies predicted by this model are within 2–7% of the experimental data. Again, these discrepancies can be explained by simplifying assumptions in the model and are accommodated by the DLAC algorithm.

To simulate damage along the truss structure, we replaced the beam elements of the third bay (highlighted in Figure 3.10) with smaller elements. Specifically, two diagonal elements were reduced in cross-sectional area by 52.7%, and two bottom elements were reduced in cross-sectional area by 63.7%. We simulated damage to the truss's numerical model by reducing the model's corresponding beam elements.

Mode	1	2	3	4	5
Analytical	19.19	38.35	63.58	66.30	90.96
Sensor 1	20.27	41.37	63.04	67.79	94.89
Sensor 2	20.28	41.40	63.17	67.89	95.08
Sensor 3	20.20	41.29	63.01	67.67	94.82
Sensor 4	20.17	41.23	63.05	67.68	94.73
Sensor 5	20.31	41.30	63.10	67.73	94.89
Sensor 6	20.23	41.29	63.02	67.68	94.81

Table 3.4: Analytical and identified natural frequencies for the damaged truss

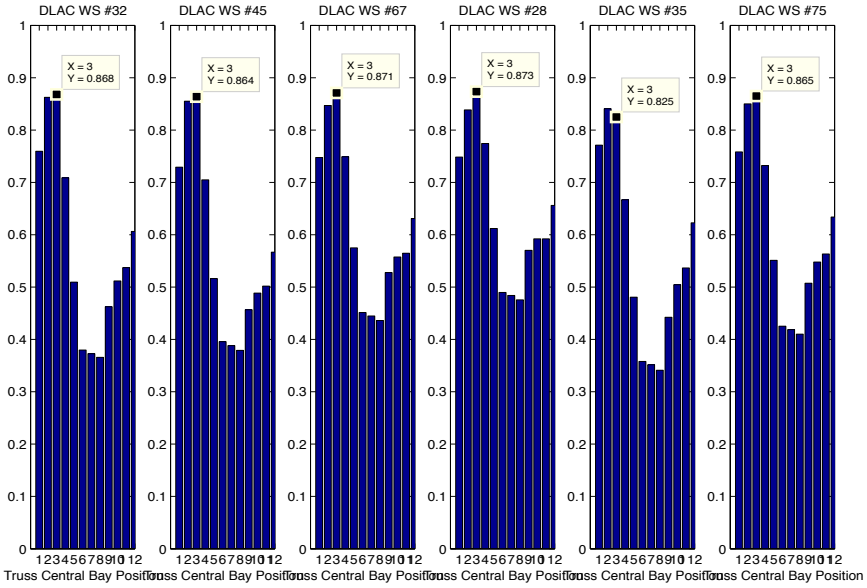


Figure 3.12: DLAC results for the damaged truss

We then excited the "damaged" truss structure and used the Imote2 nodes to collect vibration data. Because the truss has more complex behavior than the beam, we increased the sampling frequency to 560 Hz. To reduce noise, we also averaged the power spectrum results over five consecutive readings. 6 of the 11 sensors reported enough vibration data[5] to compute natural frequencies with a DLAC correlation of 85%. The natural frequency data and DLAC results are shown in Table 3.4 and Figure 3.12, respectively. The DLAC results strongly predict damage in the third bay, which is where the elements were replaced.

[5]While running the experiment, we discovered that the ITS400 sensorboard driver would occasionally deadlock while reading data; thus, nodes would fail frequently during experiments involving large amounts of data. A new sensorboard is now commercially available [68] and does not suffer from this deadlocking problem.

3.4 Evaluation: Computational Performance

We now evaluate the *computational* performance of our codesigned SHM system. First, we validate the optimal partitioning of the decentralized algorithm proposed in Section 3.2.2, by showing that it outperforms other potential partitioning points in terms of latency and energy consumption. Second, we demonstrate that our optimally-partitioned system significantly outperforms a centralized approach in terms of system lifetime.

This section considers five different configurations of our system. Four of these five configurations represent different partitionings of the decentralized algorithm discussed in Section 3.2.2: they respectively perform up to (and including) the FFT, power spectrum analysis, coefficient extraction, and equation solving stages on the mote before transmitting their partial results to the base station. The fifth configuration performs no computations and transmits its raw sensor data back to the base station, representing the behavior of a fully centralized application.

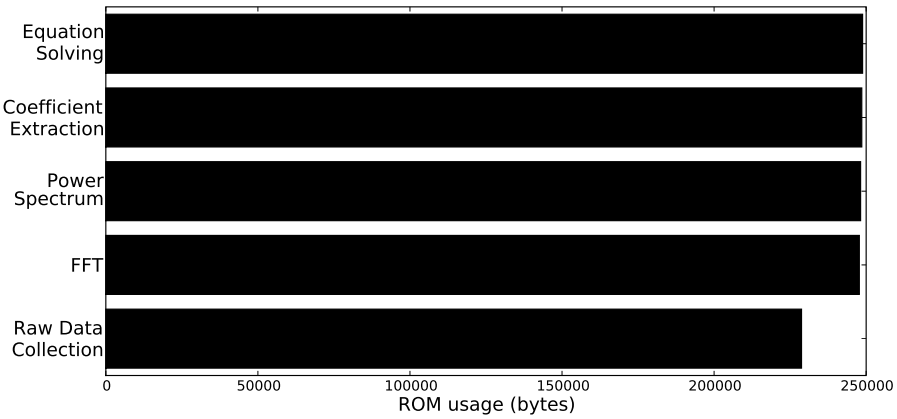


Figure 3.13: The ROM footprint of different SHM system configurations

3.4.1 Memory

Figure 3.13 presents the ROM consumption of five different configurations of our SHM system. The onboard FFT routine has the largest impact on footprint, increasing the size of the application from 228748 bytes to 247748 bytes (8.3%), while the other routines add between 264 bytes (1.1%) and 424 bytes (1.7%) each. We see a larger difference in RAM consumption as we increase the amount of onboard computation, as shown in Figure 3.14. The FFT routine again increases the footprint the most,

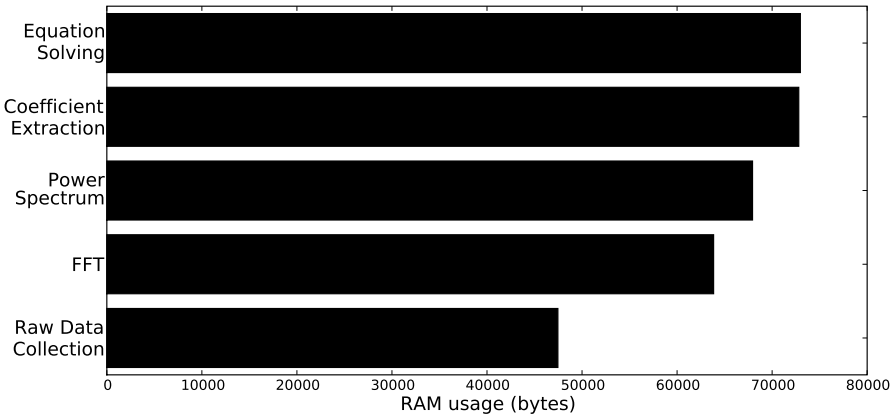


Figure 3.14: The RAM footprint of different SHM system configurations

from 47460 bytes of RAM to 63844 bytes (34.5%). The remaining routines require an additional 166 bytes (0.2%) to 4864 bytes (7.1%).

In absolute terms, this footprint fits well within the hardware capabilities of the current-generation sensor hardware. Indeed, on platforms such as the Imote2 (which is equipped with 32 MB each of flash ROM and SDRAM) this application would significantly *under*utilize the hardware capabilities. As shown above, the incremental cost of adding each additional onboard computation is also small in relative terms. Nevertheless, the memory consumption of our system could be further reduced by two straightforward optimizations, which could potentially expand the number of platforms which our system could be deployed on.

First, because our application was designed for the relatively resource-rich Imote2 platform, the codebase was not written with RAM conservation in mind. Specifically, the application retains copies in RAM of the raw sensor data and the output of intermediate computations. This decision simplifies the implementation and allows users to retrieve these intermediate values for debugging purposes. On more RAM-constrained devices, the application could be altered to keep only a single memory buffer and perform all computations in-place on this single buffer.

Second, the beta Imote2 toolchain for TinyOS 1.1 tends to greatly inflate the footprint of compiled applications compared to other platforms. The Wasabi GCC compiler used by this toolchain will crash unless the toolchain is invoked in debug mode, which disables nesC's aggressive inlining optimizations and inserts debugging symbols into the binary. Also, because binary size is not generally a concern on the Imote2, the toolchain automatically includes complex subsystems (such as a USB debugging

console) which contribute to the size of the binary. For comparison, a simple test application included in TinyOS (`CntToRfm`) consumes 195052 bytes of ROM and 18532 bytes of RAM on the Imote2 platform compared to 11234 bytes of ROM and 371 bytes of RAM for the TelosB platform. We anticipate that deploying our application with a different toolchain (whether a different platform or the more modern, stripped-down Imote2 toolchain used by TinyOS 2.1) would therefore achieve a significant footprint reduction.

3.4.2 Latency

To evaluate the latency of a single round of damage detection, we timed the execution of its constituent steps: collecting the raw sensor data from the accelerometer, performing onboard computations on the data, and transmitting the computed results back to the base station. Again, because the computation and communication latency of our SHM system depends greatly on how much computation is performed onboard, we present this data for the five different system configurations. Where possible, these latencies were measured using the Imote2's onboard microsecond timer, and are the mean of 50 rounds. Because the Imote2's onboard radio interferes with the hardware microsecond timer, the data transmission latencies (with the exception of the FFT configuration[6]) were collected over 10 rounds using an oscilloscope. The discussion here focuses on the latencies incurred by on-board processing and communication, excluding processing at the base station. It it is worth noting that this decision benefits the fully centralized approach, which pays a comparatively higher processing cost at the base station.

Figure 3.15 presents the average latency for each of these five configurations. All five schemes incur a mean cost of 3772 ms ($\sigma = 0.80$ ms) to collect raw sensor data. This closely matches the $\frac{2048}{560\,\text{Hz}} \approx 3.7$ s needed to collect 2048 samples, with some additional overhead to copy the sensor data into a local buffer. The cost of all the onboard computations is relatively small: the FFT, power spectrum analysis, coefficient extraction, and equation solving routines consume 566.8 ms ($\sigma = 2.78$ ms), 17.1 ms ($\sigma = 2.78$ ms), 97.2 ms ($\sigma = 0.01$ ms), and 27.1 ms ($\sigma = 0.26$ ms) respectively.

[6]The oscilloscope used for these measurements did not have a large enough data buffer to reliably measure the time spent transmitting the FFT data. For this configuration, it was instead measured by instrumenting the PC base station software; these results are expected to be within one packet RTT of the actual time spent transmitting.

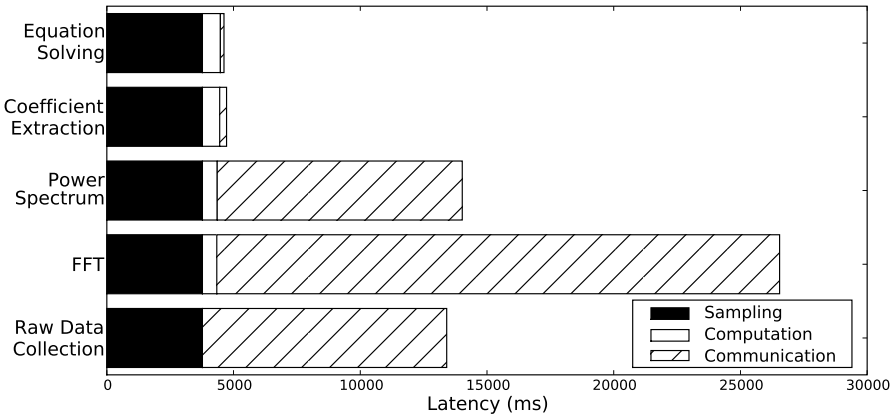


Figure 3.15: The latency of sensor data collection and processing

These latter two computations reduce the data to be transmitted by 98.8% and 99.8% respectively, from 2048 data points to 25 and 5. Therefore, these two configurations take only 271 ms (σ = 11 ms) and 142 ms (σ = 16 ms) respectively to transmit their results to the base station, compared to the 9638 ms (σ = 28 ms) to transmit all raw data in the fully-centralized case. By performing computation and an appropriate amount of processing on the nodes, very little system overhead is incurred on current-generation sensor hardware. 79.8% to 81.6% of the system's time is spent collecting data; only 20.1% or less of the latency represents reducible overhead. In comparison, the centralized approach spends 71.9% of its time transmitting data to the base station. As a result, the decentralized system can achieve latencies up to 65.5% lower than those of a centralized algorithm. It is also worth noting that delegating the equation solving substage to the base station incurs only a 2.2% performance penalty compared to doing the entire curve-fitting routine onboard, because both approaches are dominated by the time spent collecting raw sensor data. Therefore, transmitting the partial curve-fitting results is an acceptable alternative on systems where the equation solving routine cannot realistically be implemented.

Notably, performing the power spectrum analysis onboard does not reduce latency at all, and performing FFT onboard is actually counterproductive: it takes 22206 ms (σ = 133 ms) to transmit the FFT results and 9668 ms (σ = 28 ms) to transmit the power spectrum data to the base station. This phenomenon validates the data flow analysis in Section 3.2.1 (note that the single-precision floating-point values in the FFT and power spectrum data are twice the width of the 16-bit sensor readings). These findings also highlight the importance of a systematic evaluation for identifying

the optimal configuration of decentralized systems through data-flow analysis and empirical benchmarks.

3.4.3 Energy Consumption

The SHM system used during the experiments performs only limited power management, since the drivers for the Imote2 available at the time did not put all of the hardware to sleep when deactivated[7]. Nevertheless, we can estimate the energy consumption of a fully power-managing SHM system. For this analysis, an oscilloscope was used offline to measure the power draw of each of the application's hardware states (sampling, computation, and communication). These costs are then multiplied by the latency statistics given above to estimate the total energy consumption of each stage.

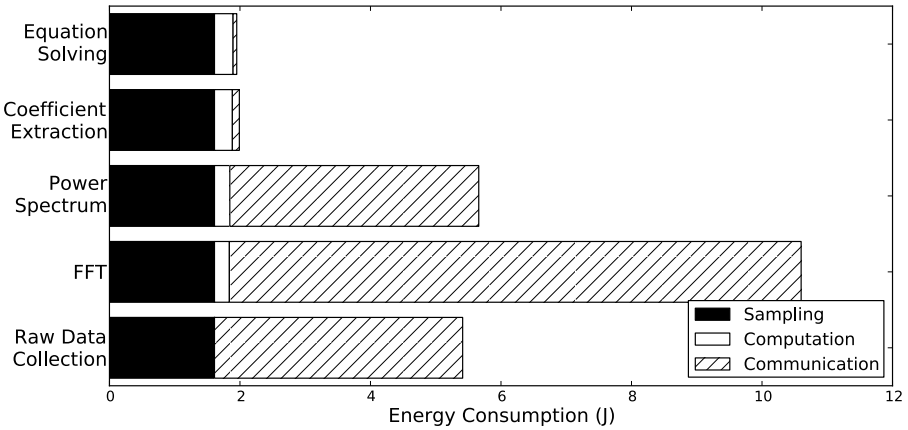


Figure 3.16: The energy consumption of sensor data collection and aggregation

Figure 3.16 shows the energy cost of a single round of SHM data collection. Performing the entire curve-fitting routine onboard compared to a fully centralized approach significant reduces the energy consumption, from 5.41 J to 1.95 J. This reduction is mainly due to the expense of sending raw sensor readings to the base station. A configuration which performs the curve-fitting routine onboard consumes 0.285 J (402 mW for 708 ms) to perform its computations. However, these computations save the node an average of 3.75 J during transmission, since it reduces the time that the radio is active and transmitting by 9.5 s. Again, offloading the equation solving portion of this routine to the base station has a minimal effect on energy consumption.

[7]A redesigned driver stack with power management functionality has since been made available through [3].

The node would save 0.011 J on computation costs but would expend an additional 0.051 J on communication, representing an increase of 2.1% compared to performing the equation solving onboard. The energy consumption of either of these two decentralized approaches is dominated by the cost of collecting raw sensor data (80.9% and 82.5% of the total energy consumption), whereas the fully centralized approach spends 70.3% of its energy transmitting the sensor readings back to the base station.

We again find that performing any fewer stages of computation onboard is counterproductive. Performing the FFT and power spectrum analysis locally incurs a computational overhead of 0.235 J but does not affect the amount of data being sent back to the base station. As a result, this approach incurs a 4.6% energy penalty compared to the fully centralized approach. Computing only the FFT data onboard performs even worse, since its output is double the size of its input. This approach therefore increases the energy consumption by 95.8% over the fully centralized case.

The memory, latency, and energy consumption benchmarks demonstrate that the optimal partitioning point indeed occurs after the curve-fitting routine, as indicated by the data-flow analysis in Section 3.2.2. These results also validate that, on systems where the full curve-fitting routine cannot realistically be implemented, even implementing a portion of this routine provides substantially better performance than simply sending the raw sensor data to the base station for processing. Again, the performance of the FFT and power spectrum routines highlight the importance of data-flow analysis in decentralized application design: in terms of RAM, ROM, latency, and energy consumption, both partially-decentralized approaches perform *worse* than a fully centralized approach.

3.4.4 Projected Lifetime

System lifetime depends on a number of factors including network configuration, duty cycle, type of power supply, and the minimum voltage at which the sensors can operate. Nevertheless, it is useful to have an approximate sense for how a SHM deployment's lifetime may be impacted by our decentralized design. Hence, in this subsection, two simplifying assumptions are made for the purposes of analysis. First, we assume a fixed energy supply of 20,250 J (the theoretical supply of 3x 1.5V, 1250 mAh AAA batteries). Second, we approximate the cost of placing the Imote2

into its deep sleep mode based on its datasheet[8] and a fixed voltage of 4.5V, since the cost is too low to measure experimentally with our oscilloscope.

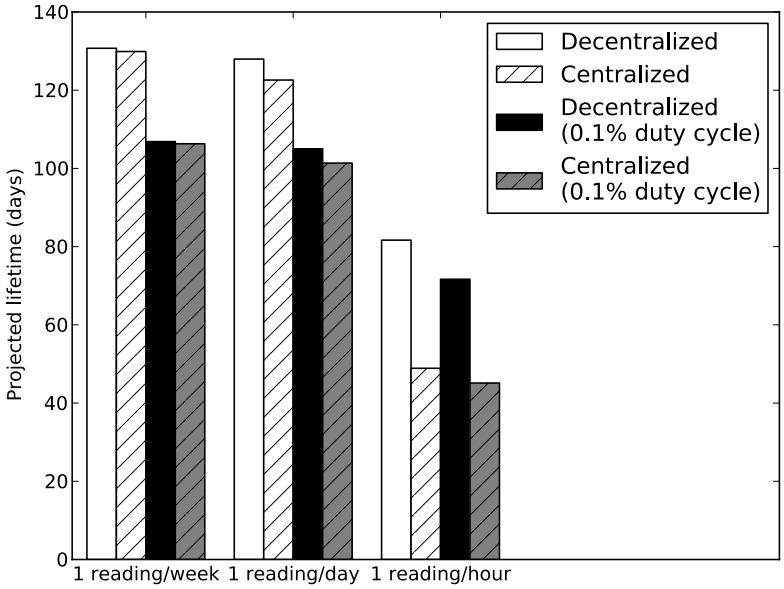


Figure 3.17: System lifetime under different usage patterns

Figure 3.17 presents the estimated system lifetime when the Imote2 is under this configuration. In the interest of reducing clutter, only the fully-centralized case (i.e., where no processing is performed onboard) and the most decentralized case (where all computations prior to the final DLAC stage are performed onboard) are presented here. As noted above, performing only the FFT or power spectrum analysis onboard would in fact reduce the node's lifetime, and running only part of the curve-fitting onboard has similar performance to the fully-decentralized case.

If the system remains asleep between periodic readings, then the decentralized approach achieves a projected lifetime of 82 days, even at a relatively aggressive hourly schedule. In contrast, the centralized approach achieves a lifetime of 49 days at an hourly schedule, though it stays within 0.6% of the decentralized approach's lifetime at lower frequencies. The sharp drop in the centralized system's lifetime occurs because sleeping dominates the system's energy cost at lower frequencies, while the high communications costs dwarf the sleeping cost at an hourly frequency. As a result, in-situ processing enables more frequent monitoring than is realistically possible for a centralized scheme.

[8]Specifically, 382 μA for the Imote2 [26] plus 15 μA for the accelerometer [77]

In practice, a SHM system may not be able to behave autonomously: its deployers may want some kind of manual control (e.g., to perform on-demand readings after a natural disaster). This can be achieved by having the nodes listen for radio transmissions between readings. Keeping the CPU and radio active at 100% duty cycles would reduce the node lifetime to only 14 hours. However, power-saving MAC layers like SCP [93] can achieve duty cycles as low as 0.1% with reasonable responsiveness tradeoffs. As shown in Figure 3.17, this would have a fairly low impact on system lifetime (a 12.2%–18.7% reduction in the decentralized case).

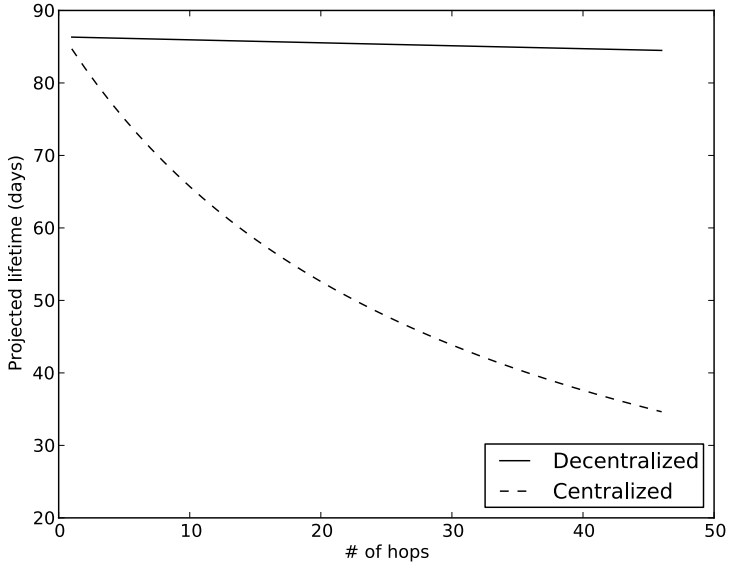


Figure 3.18: System lifetime with hourly readings and 0.1% radio duty cycle, under various network configurations

The difference in communication costs between a centralized approach and our decentralized approach are amplified under a multi-hop network configuration. This kind of network configuration is necessary for monitoring many real-world structures, since the structure's length will exceed the motes' communication range. For example, [42] required a 46-hop network to span the Golden Gate Bridge, and [18] estimates that 3–4 hops will be needed to span small bridges. The nodes closest to the sink suffer the most from communication overhead, since they must receive and relay packets from all of the nodes further away from the sink. If nodes are configured in an n-hop line, as in [42], then the node closest to the sink will have to receive $n-1$ sets of data and transmit n sets each time damage detection is performed.

As shown in Figure 3.18, under the centralized approach this node's lifetime will drop dramatically as the number of hops increases. The mote must keep its radio active

for an extra 19.3 seconds for each additional hop, transmitting during half of this time and receiving during the other half. This quickly depletes the mote's battery power, decreasing the network's lifetime from 85 days in a single-hop configuration to 77 days under a 4-hop network, and to 35 days under a 46-hop network. In contrast, the decentralized approach transmits a much smaller amount of data, so that the cost of idle sleeping still dwarfs the communication cost under any realistic hop count. A 4-hop network will reduce the decentralized system's lifetime by 3 hours, and a 46-hop network will reduce the lifetime from 86 days to 84 days. Our decentralized approach therefore represents a 9.6% increase in lifetime under a 4-hop network compared to a centralized scheme, and a 144% increase with a larger 46-hop network.

As observed in [42], reliably transporting large amounts of data over lossy links is challenging. The lifetimes of both approaches will be reduced compared to those projected here, due to packet retransmissions. However, packet retransmissions will have a significantly higher impact on a centralized system's lifetime, since its communication costs represent a much higher proportion of the total energy budget.

Chapter 4

Cyber-Physical Codesign of Distributed Structural Health Monitoring

Chapter 3 presents a distributed approach to SHM based on the Damage Location Assurance Criterion (DLAC) method. This proof-of-concept system based on DLAC represents an important first step toward deploying robust, long-lived SHM applications on real-world structures. However, DLAC has several intrinsic limitations in its SHM capabilities. First, DLAC requires the user to pre-specify the damage patterns that it should try to identify and localize. Second, DLAC is not sensitive to small damages in a structure because it only monitors the structure's natural frequencies, and because it does not correlate readings across sensors. Finally, DLAC can only properly localize damage to asymmetric structures. These limitations occur because there is effectively no collaboration among sensors under DLAC: each sensor's readings are handled independently, and are only combined at the very end to compensate for node failures and sensor noise. Alleviating these limitations requires a fundamentally new architecture which leverages collaboration among sensors to enhance the damage identification and localization results.

In this chapter, I present a hierarchical decentralized SHM system that implements a *flexibility-based* damage identification and localization method. This system represents an evolution of the co-design approach explored in the DLAC-based system, where the numerical approach is designed with both the *cyber* requirements (WSN hardware constraints) and *physical* requirements (damage detection performance) in mind. In contrast to the DLAC-based method, flexibility-based methods explicitly

correlate data across multiple sensors, allowing them to accurately identify and localize damage on a wider range of structures. The hierarchical system discussed here organizes nodes into clusters using a novel *multi-level search* approach that incrementally activates sensors in the damaged regions, allowing much of the network to remain asleep. This implementation again takes advantage of the Imote2 platform's computational power to perform in-network processing wherever possible; thus, nodes further save energy and bandwidth by only transmitting the intermediate results related to the flexibility calculation.

Specifically, this chapter presents the following contributions:

1. a *cyber-physical architecture* which efficiently maps flexibility-based damage identification and localization methods onto a distributed WSN;
2. an *implementation* of this architecture on top of the TinyOS operating system [1] and ISHMP services toolsuite [3]; and
3. *empirical evaluation* of this implementation on simulated and real truss structures.

Experimental results demonstrate that this flexibility-based approach can successfully localize damage on both structures to the resolution of a single element. Latency and power consumption data collected during these experiments also demonstrate the energy efficiency of this codesign approach.

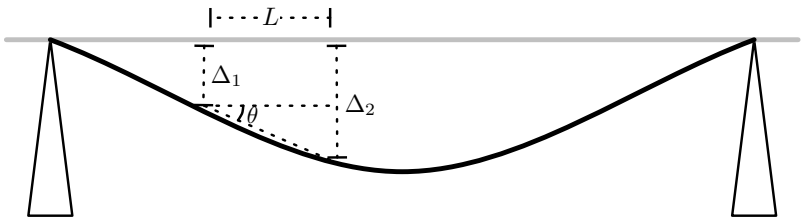


Figure 4.1: Structural deflection

4.1 Damage Localization Approach

This section introduces the *physical* (structural engineering) aspects of our decentralized damage localization system. This system is based on a family of damage

localization techniques collectively known as *flexibility based* algorithms. The intuition behind these methods is that structures will flex slightly when a force is applied, as shown in Figure 4.1. As a structure weakens, its stiffness decreases, and thus its flexibility changes. Changes in structural flexibility over a structure's lifetime can be used to identify and localize damage [65].

This subsection provides a brief background on two particular flexibility-based methods used within the decentralized system. While flexibility-based methods are well-known in structural engineering literature, the existing research generally deals with algorithmic issues (i.e., selecting the best numerical methods for damage identification and localization) rather than efficiently deploying these methods on a distributed architecture for WSNs. As with DLAC, the underlying numerical methods discussed in this subsection are existing work which are not part of this dissertation's contribution. I focus here on the details of these algorithms that are most relevant to system design; more mathematical details can be found in [28, 91].

Flexibility-based methods are executed in two stages. When the system is first turned on, a baseline structural modal identification is performed. The sensors simultaneously collect vibration data. Multiple sensors' data are correlated to identify the structure's *modal parameters* (natural frequencies and mode shapes). The modal parameters are then further processed to compute the structure's *flexibility matrix*.

Online, the data collection and processing phases above are repeated, and the base station produces a new flexibility matrix. By subtracting the new flexibility matrix from the stored one, the base station can determine if the structure is damaged (and if so, identify the damaged region).

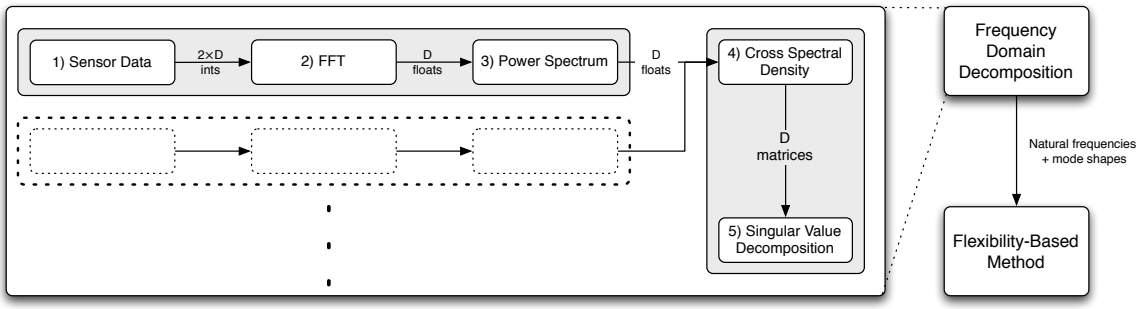


Figure 4.2: The data flow of a traditional flexibility-based method

The main components of flexibility-based methods are illustrated in Figure 4.2. The structure's modal parameters are identified using Frequency Domain Decomposition

(FDD), an existing structural engineering technique which can be decomposed into several stages. Traditionally, FDD is executed as follows. (1) All the nodes in a cluster simultaneously collect D vibration samples using their onboard accelerometers. The size of D depends on structural properties (like its complexity and material) as well as the modes we are interested in, and is typically hundreds or thousands of samples. (2–3) Each node independently performs an FFT and power spectrum analysis on the vibration data, transforming it into magnitudes in the frequency domain. (4) D magnitudes collected from each node are correlated to compute a Cross Spectral Density (CSD) matrix. (5) A Singular Value Decomposition (SVD) is performed on the CSD matrix at each of D discrete frequencies. The singular value in each singular value matrix is collected to form a vector, and the structure's P lowest natural frequencies are identified as the peaks in this vector. The mode shapes corresponding to the natural frequencies can be estimated from the first column of the corresponding left SVD matrix.

The FDD output is then input into a flexibility-based method. The system described in this chapter uses two specific flexibility-based methods: the Angles-Between-String-and-Horizon flexibility-based method (ASHFM) [28] and the Axial Strain flexibility-based method (ASFM) [91]. These two methods are particularly compelling because they can localize damage down to a resolution of a specific element on beam-like and truss-like structures, respectively. Most other flexibility-based methods localize damage only to less specific regions of the structure, while [8] achieves similar damage localization resolution at a much higher computational cost.

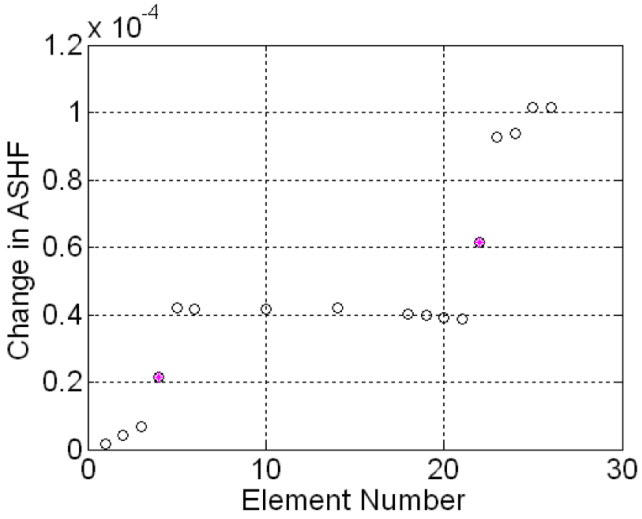


Figure 4.3: Example ASHFM damage indicator output; shaded dots correspond to damaged elements

ASHFM measures the flexibility of a beam-like structure as the angle θ in Figure 4.1. The FDD output data is used to calculate θ at each of the structure's modes, producing a flexibility matrix F. The difference $\Delta F = \left|F^b - F\right|$ can be used to localize the changes in flexibility to locations along the beam, where F^b is the flexibility matrix calculated during the baseline phase. Specifically, the maximum absolute values of the components in each column or diagonal of ΔF are extracted as *damage indicators*. When a location on the structure is damaged, it will appear as as "step and jump" in the plot of damage indicators, as shown in Figure 4.3. In this example, the large jumps in the ASHF damage indicator surrounding the shaded points reflect damage in the corresponding structural elements.

ASFM achieves damage localization at a similar resolution to ASHFM, except on truss-like rather than beam-like structures. At a high level, ASFM localizes damage in much the same way as ASHFM. The main differences are that ASFM requires vibration data taken simultaneously in multiple directions, and that the two methods have different formulations for computing F.

4.2 Distributed Architecture

The numerical methods discussed above have been designed with centralized networks in mind, where sensors are used as simple data collection devices that can stream large data sets to a central server over a wired backbone. Under a WSN, this approach is inappropriate because of the nodes' limited network and energy resources. However, in order to design an efficient decentralized architecture, we can leverage a particularly powerful feature of these flexibility-based methods. Specifically, they enable a tradeoff between energy consumption and localization resolution: the more nodes that are activated, the finer-grained the damage localization.

We leverage this feature to construct an energy-efficient, *multi-level* damage localization system which selectively activates additional sensors at each level in order to more precisely localize structural damage. In the common case that the structure is not damaged at all, only a minimal subset of nodes are enabled, considerably reducing the system's energy and bandwidth consumption. This approach naturally maps to a hierarchical, cluster-based distributed network architecture. In addition, to promote a more efficient mapping onto our distributed system, we leverage an existing *peak picking* technique to reduce the data flow among sensors participating in each cluster.

4.2.1 Multi-Level Damage Localization

Although adding more sensors can improve a flexibility-based method's localization results, only a handful of sensors are needed to accurately *identify* damage. In the first stage of the multi-level search, this minimal number of sensors are enabled, forming a single cluster. Damage identification and localization is performed using this small subset of sensors. In the common case that no damage is identified, the search ends and all the nodes return to sleep.

In the event that damage is identified, the flexibility-based method will also output coarse-grained damage localization. For example, ASHFM will identify two adjacent sensors surrounding each damage location on the structure. In the next round of the multi-level search, the system activates additional sensors in the region of interest and repeats the entire procedure, including collecting new vibration data. This second round subsequently localizes the damage to a smaller region than the first round. The system may repeat this drill-down procedure to achieve even finer grained results until the desired resolution is reached.

The key feature of this approach is that it does not activate the entire sensor network at once. Instead, relatively few sensors are used to identify damage; and when damage is identified, only those sensors in the area of interest are incrementally added to the search. As a result, many nodes are able to remain asleep for part or all of the multi-level search. This approach will also scale to larger structures, since the cost of the search is no longer proportional to the size of the structure. As discussed in Section 4.3.2, the reduced energy burden can also be distributed across the network by activating different subsets of the network at different times.

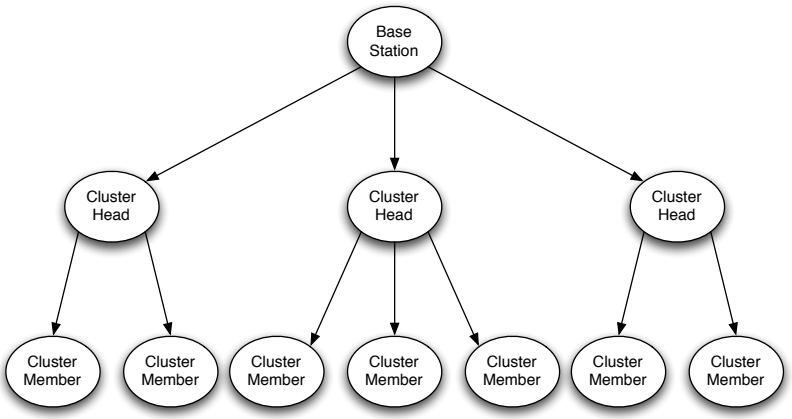


Figure 4.4: Sensor Roles in the System

4.2.2 Network Hierarchy

Once the nodes participating in this multi-level search are selected, they are each assigned one of three different roles: *cluster member*, *cluster head*, and *base station*. A node's role determines what data it handles as well as its level in the network hierarchy, as shown in Figure 4.4. To allow the system to better scale to large structures, the nodes may be organized into multiple independent clusters. Each cluster operates as an independent unit, with the cluster head coordinating nodes within its cluster and ultimately transmitting the cluster's (relatively small) mode shape data to the base station for final processing.

Based on these roles, the system operates as follows. The cluster members collect raw vibration samples from their onboard accelerometers. They then carry out an FFT to transform the vibration response into frequency domain data, followed by a power spectrum analysis.

The cluster head nodes aggregate the extracted power spectrum data from the cluster members beneath them in the hierarchy. There, the CSD and SVD are carried out to extract the structure's mode shape vector.

The cluster heads then transmit the mode shapes to a single base station node, which calculates the structure's flexibility. The flexibility is then used to identify and localize any structural damage.

4.2.3 Enhanced FDD

Efficiently implementing this architecture for a flexibility-based system is challenging because there are no obviously "best" places to introduce network communication: the CSD and SVD routines are necessarily computed on a single node with access to all the other cluster members' data, and the prior steps all have very large outputs (hundreds or thousands of points). In order to achieve truly energy-efficient behavior, we must optimize the FDD algorithm's data flow to promote an efficient mapping onto wireless sensor networks.

We leverage an optimization proposed in [84,99] that adds a new *peak picking* stage to FDD. To illustrate how this optimization works, note that most of the computations in the FDD routine do not contribute to the final results. As described in Section 4.1,

the CSD step normally requires the cluster head to pool D data points from each of its cluster members. This data is processed into D CSD matrices, which the SVD routine further processes into D outputs and discards all but the P corresponding to the structure's natural frequencies (note that $P \ll D$). A key observation about this procedure is that the ith CSD matrix is only constructed using the ith power spectrum data point from each cluster member. Moreover, only the P CSD matrices corresponding to the structural's natural frequencies contribute to the FDD stage's final output.

The peak picking routine allows each node to independently identify these P natural frequencies solely from local data. Hence, only those P relevant data points are passed onto the CSD stage, which in turn passes only the relevant P matrices onto the SVD stage. In this way, both the computational and communication cost of identifying modal parameters are reduced considerably. Again, the data which the nodes withheld would not have contributed to the final flexibility computation. Hence, even though significantly fewer data are transmitted and processed, there is no loss in damage identification or localization performance.

4.3 Implementation

We have built a proof-of-concept implementation of our system on top of the Imote2 [26] sensor platform using the TinyOS operating system [1]. This implementation utilizes the ISHMP services toolsuite [3] developed by the Illinois Structural Health Monitoring Project (ISHMP), which provides subsystems for sensor data acquisition, reliable data transmission, remote procedure calls, and time synchronization based on the FTSP protocol [58].

4.3.1 Hardware Platform

The Imote2 is an advanced wireless sensor node platform built around the low-power PXA271 XScale processor and 802.15.4-compliant radio hardware (Chipcon CC2420) with a built-in 2.4GHz antenna. While distributed architecture discussed in this chapter is not inherently tied to a particular platform, the Imote2 offers several salient

improvements over previous generation WSN platforms that are particularly useful for our application.

First and foremost, the PXA271 CPU has 256 KB of embedded SRAM and can address 32 MB of on-board SDRAM, providing copious space for computations. In contrast, platforms such as the TelosB [66] and MICA [23, 24] family have access to only 4–10 KB of RAM, which would not even be enough to store the entire raw sensor reading dataset. Accordingly, such platforms would either be restricted to purely streaming computations or would have to swap data in and out of onboard flash, a potentially expensive operation.

Second, the PXA271 CPU can be dynamically clocked from 13 – 416 MHz, allowing nodes to increase their CPU speed when needed (e.g., while collecting high-resolution sensor data) and decrease its speed at other times to save energy. Third, the Imote2 is a modular stackable platform which can be expanded with extension boards to customize the system to a specific application. The SHM-A [68] sensor board provides an add-on accelerometer which we have confirmed to be sufficiently accurate for our SHM application. Fourth, the Imote2 is equipped with 32 MB of flash memory, which allows us to deploy the entire application on all nodes in the network. We take advantage of this capability to dynamically reconfigure the network without having to re-flash the nodes with new software, as discussed below.

4.3.2 Software Platform

As described above, the system is implemented in the nesC programming language on top of the TinyOS 1.1 operating system. Several major components from UIUC's ISHMP Toolsuite 3.0 were leveraged to ease implementation. Specifically, ISHM's `RemoteCommand` RPC subsystem is used to reliably coordinate motes and collect partial results, and the `SensingUnit` components are used to start data collection simultaneously across all the participating motes.

Figure 4.5 illustrates the network configuration process. At the start of the procedure, the base station constructs a configuration packet containing information about the cluster division and the assignment of roles within each cluster. The base station disseminates this packet to the cluster head, which in turn disseminates it to the other cluster members. Because the Imote2 platform is equipped with copious flash

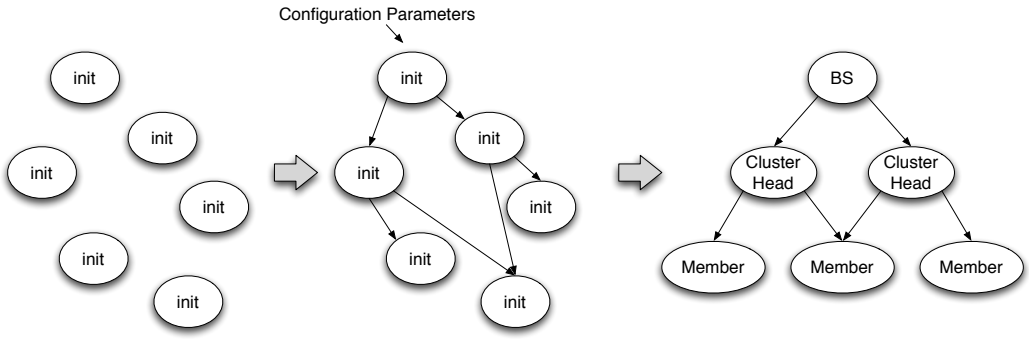


Figure 4.5: Network Configuration Process

memory, all nodes are programmed with the code for all roles. Thus, nodes can handle the reconfiguration message by simply changing their configuration parameters in RAM.

After all nodes are configured, the base station disseminates a control packet into the network to start data collection; this message propagates to the cluster heads and cluster members in a similar fashion to the configuration packet. After completing data collection and computation, cluster members deliver the partial results to the cluster heads. Similarly, the cluster heads deliver mode shape data to the base station after completing their computations. Rather than implementing the complex final damage calculations directly on the base station, the base station outputs the cluster heads' data over its serial port. This output is collected at a PC attached to the mote, and the final computations are performed in MATLAB. If damage is identified, more fine-grained damage localization may be triggered as discussed in Section 4.2.

Due to time constraints, the current implementation is not yet fully automated. The user must manually trigger each level in the multi-level search, inputting the desired network configuration based on the previous level's damage detection results. Moreover, the base station does not currently allow users to configure networks containing more than one cluster. These are not fundamental limitations of the architecture, and could be lifted with additional base station code.

4.4 Evaluation

To validate our system, we implemented and deployed our multi-level damage localization system on a simulated steel truss structure, which presents a challenging scenario

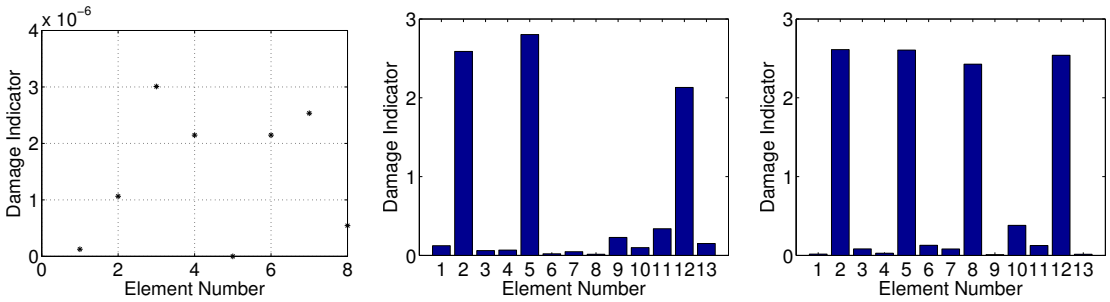
for damage localization due to its structural complexity. Experimental results demonstrate that our system is able to accurately localize damage at the member-level to both structures. Moreover, latency and energy consumption data collected during the truss structure experiment illustrate the efficiency of our decentralized approach. Preliminary results also validate the approach on a real truss structure.

4.4.1 Simulated Truss

Where not otherwise noted, the experiments in this section involve simulated sensor data from a 5.6 m steel truss structure [21] at the Smart Structure Technology Laboratory (SSTL) at the University of Illinois at Urbana-Champaign. In order to accommodate the truss's increased structural complexity, we increased the sampling frequency to 560 Hz, the record length to 18,432 data points, and the FFT size to 4096 points. We attempted to carry out the experiment using an earlier implementation of the damage localization system based on the Crossbow ITS400 sensorboard. However, as noted earlier in Section 3.3.2, the ITS400 sensor subsystem contains a serious deadlocking bug which prevented us from collecting sufficient vibration data to perform our experiments on the real truss.

Instead, we produced two sets of simulated data traces using a finite element model of the truss in MATLAB, with additional measurement noise added to simulate noisy sensor readings. The first set represents the truss in its intact case, providing a baseline flexibility measurement. The second set was generated with simulated damage to three members of the left side of the truss and four members to the right side of the truss.

For the truss experiments, we wished to evaluate our system's damage localization performance as well as its energy consumption. Thus, we made two augmentations to our nesC code for this set of experiments. First, we added a "fake" sensor driver which replayed sensor data traces from the motes' flash memory, allowing us to inject our simulated traces into live experiments. Second, we collected timestamping data at key points in our code in order to measure the latency and energy consumption of each major component of our system.



(a) level-1 identification (b) level-2 localization (left) (c) level-2 localization (right)

Figure 4.6: Damage localization results on the simulated truss

Damage Localization

To evaluate our system's damage localization performance, we performed three different experiments with nine Imote2 motes. In our first configuration, we injected simulated sensor data collected at uniform points along the truss's length. This configuration represents the "level 1" damage identification phase. The damage indicators computed during this experiment are plotted in Figure 4.6(a). Based on the step-and-jump surrounding bays 4 and 10 (shown as the two peaks in Figure 4.6(a)), our system correctly identified damage on both halves of the truss.

In our second and third configurations, we used simulated sensor data collected at a greater density on the truss's left and right halves, respectively. This configuration represents the more fine-grained "level 2" damage localization phase. As shown in Figures 4.6(b) and 4.6(c), our system indeed correctly localized the three damaged members on the left side of the structure and the four damaged members on the right side.

Energy Consumption

During the experiments described above, we collected timestamp data from the motes in order to directly measure the latency of each major stage in the experiment. We also performed a separate set of experiments to measure the latency of time-synchronizing the motes and collecting 18,432 data samples (since the previous truss experiments used replayed data traces). Tables 4.1 and 4.2 present the average latencies for the cluster member and cluster head nodes, respectively. Offline, we measured the power

draw of each stage using an oscilloscope, which we used to estimate the total energy consumption of each stage in the experiment.

State	Latency (s)	Energy (J)
Synchronization	30.00	12.06
Sensing	53.80	22.96
Compute FDD	21.47	9.28
Transmit FDD	0.21	0.08

Table 4.1: Mean latency and energy cost at cluster member

State	Latency (s)	Energy (J)
Synchronization	40.35	16.23
Sensing	49.68	21.20
Compute own FDD	18.19	7.86
Receive other FDD	0.56	0.23
Compute mode shapes	1.52	0.66
Transmit mode shapes	1.35	0.53

Table 4.2: Mean latency and energy cost at cluster head

Several important observations can be made from this data. First, our decentralized architecture is indeed effective at dramatically reducing the amount of bandwidth and energy consumed in exchanging data among nodes. Our decentralized architecture spends an average of 0.21 s per cluster member exchanging FDD results, plus an average of 1.35 s per cluster head transmitting the mode shape results to the base station. In contrast, based on the analysis in Section 3.4, we estimate that it would have taken 87 s per sensor to reliably transmit the 18,432 raw sensor readings to the base station for centralized processing.

Second, our efficient architecture incurs relatively little overhead on the Imote2 hardware. On the cluster member nodes, as much as 79.4% of the latency and 78.9% of the energy consumption can be attributed to synchronizing the nodes and collecting data. Only 21.1% of the energy consumption represents reducible overhead. The cluster head nodes incur similarly low overheads, with only 20.4% of the latency and 19.1% of the energy consumption attributable to processing and data transmission.

Third, this low overhead leads to low total energy consumption in absolute terms. On average, the cluster member and cluster head nodes consume a total of 44.4 J and 46.7 J, respectively. A typical power supply of 3x 1.5V, 1250 mAh AAA batteries delivers a theoretical energy supply of 20,250 J. Thus, with proper duty cycling, we

anticipate that each node could perform damage localization hundreds of times before depleting its energy supply.

4.4.2 Real Truss

Since the simulated truss experiments, we have reimplemented the system using a newer version of the ISHMP library and the SHM-A sensorboard (which, crucially, does not suffer from the deadlocking bug experienced with the ITS400 sensorboard). Using this new implementation, we have carried out preliminary experiments of two-level damage localization on a full-scale (17.04m L × 1.83m W × 1.98m H) steel highway truss at Purdue University.

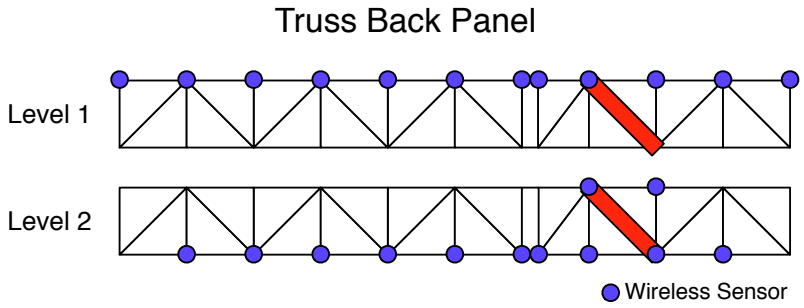
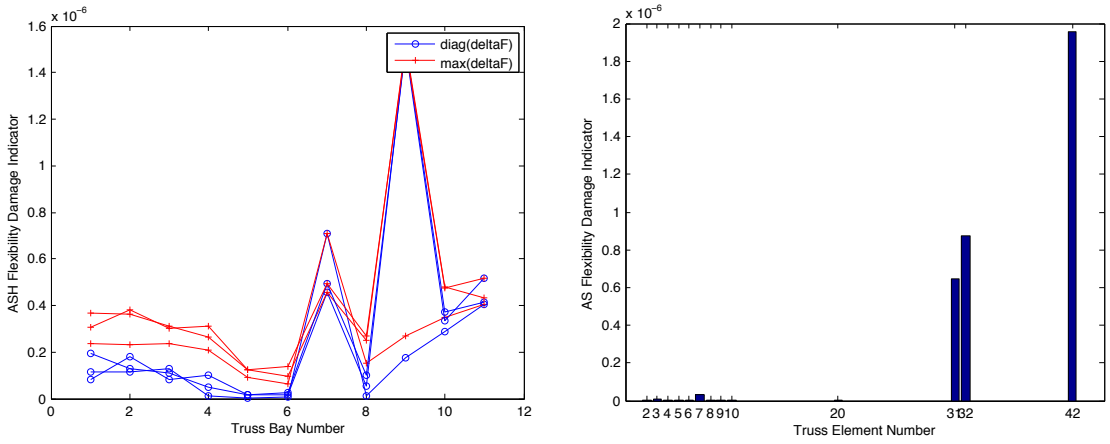


Figure 4.7: Truss experimental setup; highlighted element was damaged by cutting halfway through



(a) Level 1 damage localization results

(b) Level 2 damage localization results

Figure 4.8: Damage localization results on the real truss

We obtained mode shapes from the truss in different configurations, while varying the level of damage and the number of damage locations. Figure 4.8 presents the output

from a successful experimental run, corresponding to the configuration illustrated in Figure 4.7. After collecting data from the intact truss, the truss was damaged by cutting halfway through a diagonal beam on the back panel of the truss's ninth bay. For the first level of damage localization, twelve sensors were deployed along the top of the truss's back panel. Each sensor collected 65,536 data points at a frequency of 280 Hz; aggregated data were collected at the base station, where ASHFM was employed to localize damage to the resolution of a bay. These level 1 results, plotted in Figure 4.8(a), correctly identify damage at the ninth bay.

Based on these results, ten sensors were redeployed on the bottom of the truss's back panel, along with two on the top surrounding the damaged bay. New data were collected at the same sampling rate and size for a second level of damage localization, and ASFM was used at the base station to localize damage to a specific element. These level 2 results, plotted in Figure 4.8(b), correctly localize damage to element 42. Both levels of damage localization match simulation results and the actual damage scenario, validating both the numeric approach and its distributed implementation.

Although these results validate the underlying numerical and computational methods, the complexities of real structures still pose significant system challenges. Inconclusive results may result when noise in the raw sensor data is not averaged out, or when the damage level is very small. Simulation in MATLAB has shown that intermediate averaging of the results can effectively improve the data's signal-to-noise ratio. Further examination of the data will be performed to better understand the limitations and capabilities of the damage detection technique for this particular structure.

Chapter 5

Clinical Early Warning and Real-Time Event Detection

Within the medical community, there has been significant research into preventing clinical deterioration among hospital patients. Clinical study has found that 4–17% of patients will undergo cardiopulmonary or respiratory arrest while in the hospital [80]. Early detection and intervention are essential to preventing these serious, often life-threatening events. Indeed, early detection and treatment of patients with sepsis has already shown promising results, resulting in significantly lower mortality rates [39].

Two distinct approaches have emerged to support these early detection efforts. First, existing electronic medical records may be processed using machine learning techniques to identify early warning signs of clinical deterioration [81]. Second, real-time data collection systems based on WSN technologies have been proposed to collect vital sign data at a high granularity [20, 46]; in turn, this data may be used as inputs to automated scoring systems [41].

The application of either approach in isolation has significant limitations. Early warning systems based on existing medical records suffer from the sparseness of measurements: in general hospital units, patients' vital signs are typically collected manually by a nurse, at a granularity of only a handful of readings per day. While real-time monitoring systems fill the gap in measurements, it is nevertheless impractical to intensively monitor *all* patients in general hospital units. Such an approach would introduce issues of scalability and complexity; it would also inconvenience many patients, who would be asked to wear sensors even if they were not necessary.

This chapter presents work towards a novel *two-tier* system for clinical early warning and intervention. This system combines the two approaches in order to achieve the

benefits of both (identifying clinical events) while avoiding their individual shortcomings. At the first tier, advanced machine learning techniques are used to identify at-risk patients from real-time data in existing electronic medical records. At the second tier, these at-risk patients are provided WSN-based monitoring devices which collect high-fidelity vital sign data in real time. These tiers are joined by a predictive algorithm which scores a patient's predicted outcome based on *both* real-time vital sign data and the coarser-grained data found in electronic medical records.

The envisioned two-tier system is a long-term, multidisciplinary effort with many open research problems beyond the scope of this dissertation. This chapter discusses my contributions to two components of this effort. Section 5.2 describes a recent study which applied machine-learning techniques to historical clinical data, in order to identify patients most at risk of being transferred to the ICU. The results of this study represent a proof-of-concept for the first tier of the envisioned system, and are currently being used to select candidates for a year-long clinical trial of real-time vital sign monitoring at Barnes-Jewish Hospital. Section 5.3 discusses the ongoing clinical trial in further detail, focusing on the important lessons learned while deploying a WSN for clinical monitoring at this scale. The data collected during this study will ultimately be used to guide the development of the system's second tier.

5.1 Related Work

Numerous projects in existing literature propose using inexpensive wireless hardware to assist in healthcare. Most closely related are CodeBlue [54], Alarm-NET [88], MEDiSN [46], and a pilot deployment at Barnes-Jewish Hospital [20], which deploy WSN-based devices for real-time patient monitoring. A common theme among these systems is that they focus primarily on systems issues related to collecting raw sensor data in a wireless sensor network: hardware design, network protocols, and system reliability. This chapter discusses work towards a *complete* system for clinical event detection; real-time vital sign data would be mined along with existing electronic medical records to predict patient outcome. The real-time data collection component of this system leverages the contributions made by these prior studies. Moreover, the experiences discussed in Section 5.3 provide unique insights into deploying large-scale data collection systems in a clinical setting.

The development of algorithms for detecting clinical deterioration has a long tradition in medical literature. Two types of algorithms for detecting clinical deterioration may be distinguished: algorithms that rely on medical knowledge and general machine learning techniques that do not require domain knowledge. A number of scoring systems that use medical knowledge exist for various medical conditions. For example, Yandiola [92] evaluates the effectiveness of Severe Community-Acquire Pneumonia (SCAP) and Pneumonia Severity Index (PSI) in predicting outcomes in patients with pneumonia. Similarly, outcomes in patients with renal failure may be predicted using the Acute Physiology Score (12 physiologic variables), Chronic Health Score (organ dysfunction), and agE (APACHE) score [45]. However, these algorithms are best suited for specialized hospital units. In contrast, the detection of clinical deterioration on general hospital units requires more general algorithms. For example, the Modified Early Warning Score (MEWS) [41] uses systolic blood pressure, pulse rate, temperature, respiratory rate, age, and BMI to predict clinical deterioration. These physiological and demographic parameters may be collected at bedside, making MEWS suitable for general hospital units.

An alternative to algorithms that use medical knowledge is to adopt standard machine learning techniques. This approach has two important advantages over traditional rule-based algorithms. First, it allows us to consider a larger number of parameters during the prediction of patient outcomes. Second, since they do not use a small set of rules for predicting outcomes, it is possible for machine learning approaches to achieve improve accuracy. Learning techniques such as decision trees [81], neural networks [95], and logistic regression [35] have been used to identify clinical deterioration.

5.2 Early Warning System Design

The first step in providing early intervention lies in determining which patients are most at risk of clinical deterioration and who would most benefit from more intensive monitoring. In order to make this decision, our proposed system will leverage existing data from electronic medical record systems. As hospitals transition to electronic record-keeping, their databases are aggregating an increasing wealth of information about a patient's history and current condition: demographic data, the results of lab tests, information about prescriptions, low-granularity vital sign data, etc.

The challenge lies in determining how all the factors represented in this data may serve as potential warnings of clinical deterioration. We envision the use of advanced machine-learning techniques to find correlations between medical data and the patient's future outcome. These correlations can then be used to identify at-risk patients based on the data contained in their medical records.

In order to demonstrate the feasibility of an Early Warning System component, we applied *logistic regression* [37] techniques to predict patients' outcomes (specifically, whether or not they would be transferred to the ICU) on a historical dataset. This dataset cataloged 28,927 hospital visits from 19,116 distinct patients between July 2007 and January 2010. It contained a wide array of demographic and medical data for each of the visits, such as age, manually-collected vital sign data, pharmacy data, and whether or not the patient was transferred to the ICU. All data contained in this dataset were taken from historical EMR databases, and reflects the kinds of data that would realistically be available at the first tier of our proposed clinical warning system. This study serves as a proof-of-concept for our vision of using machine learning to identify at-risk patients and (ultimately) to perform real-time event detection.

Another important challenge in implementing clinical early warning is generating a manageable number of alarms. If alarms are too frequent, clinicians may be more inclined to ignore them. A key feature of logistic regression is that it allows a trade-off between sensitivity and alarm rate which may be adjusted in deployment. As discussed below, performance results show that this approach achieves acceptable predictive performance even under low alarm rates.

In this section, we first provide a background description of the logistic regression approach used in this study. We then describe the performance of this approach under two different scenarios: a retrospective analysis where each patient is assigned a single score, and a real-time simulator where patient scores are continually recomputed as new EMR data is entered.

5.2.1 Algorithm Overview

Background

At a high level, logistic regression predicts an outcome y from a vector of input variables $x_1, x_2, \ldots, x_k$. Each element in the vector x represents one reading of a particular kind; for example, x_1 may indicate a patient's blood oxygen saturation while x_2 may be the patient's temperature. The predicted outcome $y \in (0, 1)$ indicates the likelihood of some event occurring based on the input variables. y values near 1 represent a strong prediction of the event occurring (e.g., the patient being admitted to the ICU), while values near 0 represent a high likelihood of the event not occurring. The variable y is discretized using a simple thresholding procedure. The threshold controls the trade-off between false positive and false negative rates (i.e., higher thresholds decrease false positives but increase false negatives).

In order to compute predictions, a set of weights $\beta_1, \beta_2, \ldots, \beta_k$ are used to determine the importance of each parameter that is part of vector x. Large positive β values indicate that a particular variable is highly predictive of an outcome of interest, while large negative β values indicate that a variable is highly protective against this outcome. For example, when x_1 represents a patient's blood oxygen saturation, a large negative β_1 would indicate that patients with higher blood oxygenation levels are less likely to be ultimately admitted to the ICU.

The logistic equation is formally defined as follows. For each case (hospital visit), we have an input vector $x_1, x_2, \ldots, x_k$. We also have a set of coefficients $\beta_1, \beta_2, \ldots, \beta_k$ and an intercept value β_0 which are constant across all cases. From these inputs, we may compute

$$z = \beta_0 + \sum_{i=1}^{k} \beta_i x_i$$

for each case. The output z is in the range $(-\infty, \infty)$. We then confine the prediction to the range $(0, 1)$ by computing

$$y = \frac{1}{1 + e^{-z}}$$

The vector β is produced offline from a training (historical) dataset. Each case in the training dataset has both an input vector x and a known outcome y. Since the

outcome is discrete and known, y may only take the values of 1 (positive outcome) or -1 (negative outcome). β may then be fitted to the training dataset using standard techniques, such as least-squares.

Time Division

By itself, logistic regression does not operate on time-series data. That is, each variable input to the logistic equation corresponds to exactly one data point: e.g., a blood pressure variable would consist of a single blood pressure reading. In a clinical application, however, it is important to capture unusual changes in vital sign data over time. Such changes may precede clinical deterioration by hours [13], providing a chance to intervene if detected early enough. In addition, not all readings in time-series data should be treated equally; the value of some kinds of data may change depending on its age. For example, a patient's condition may be better reflected by a blood-oxygenation reading collected one hour ago than a reading collected 12 hours ago.

To capture the temporal effects in our data, we retain a sliding window of all the collected data points within the last 24 hours. We then subdivide this data into a series of n equally-sized buckets (e.g., 6 sequential buckets of 4 hours each). In order to capture variations within a bucket, we compute three values for each bucket: the minimum, maximum, and mean data points[9]. Each of the resulting $3n$ values are input to the logistic regression equation as separate variables.

Handling Missing Data

A complication of using clinical data is that not all patients will have values for all variables. Many types of clinical data involve lab tests that are not routinely performed on all patients. This problem is compounded by dividing time into buckets: even when a patient has had a particular lab test, it will only provide a data point for one bucket.

[9]We attempted to add a fourth value for standard deviation, but found that our historical data set was too sparse to calculate σ for nearly all buckets. We are considering other ways to measure variability as future work.

To deal with missing data points, we first "carry over" historical values. Specifically, when a bucket is empty, we insert the patient's most recent reading from any time earlier in the dataset, possibly going all the way back to the beginning of the hospital stay. To handle cases where a patient had *no* previous data for a particular variable, we also calculate the mean value for each variable over the entire historical dataset. These mean values are used as a fallback when the carrying-over procedure cannot find any data to carry over.

5.2.2 Performance Results

Retrospective Analysis

After implementing the above algorithm in MATLAB, we evaluated its accuracy over the historical dataset. Specifically, logistic regression was used to predict whether the patient was admitted to the ICU.

For the purposes of training, we looked at a single 24-hour window of data from each patient. For patients admitted to ICU, this window consisted of 26 hours – 2 hours prior to ICU admission; for all other patients, this window consisted of the first 24 hours' of their hospital stay. We subdivided this 24-hour window into 6 contiguous windows of 4 hours each.

Variable	**Coefficient**
Respirations (bucket 6 max)	4.45
Oxygen Saturation, pulse oximetry (bucket 6 min)	−4.22
Shock Index (bucket 6 max)	4.01
Respirations (bucket 6 mean)	3.41
BP, Systolic (bucket 6 min)	−2.96
Coagulation modifiers (bucket 1)	2.70
Pulse (bucket 6 max)	2.55
Respirations (bucket 6 min)	2.51
BP, Diastolic (bucket 6 max)	2.48
Oxygen Saturation, pulse oximetry (bucket 6 mean)	−2.44

Table 5.1: The 10 highest-weighted variables from the training dataset

As noted above, the logistic model requires training data to fit the unknown β values from known data and outcomes. We divided the dataset into two halves; the first half was used to train the model. The dataset's 36 categories were then divided into

buckets and min/mean/max features wherever applicable, resulting in 398 variables. Table 5.1 provides a sample of output from the training process, listing the 10 highest-weighted variables and their coefficients; all 398 variables were used to predict patient outcomes. We then used the second half of the dataset as testing data. We generated a predicted outcome for each case in the testing data, using the β weights derived from the training data. As noted above, we then applied various thresholds to convert these predictions into binary values, and compared the results against the ground-truth ICU outcome.

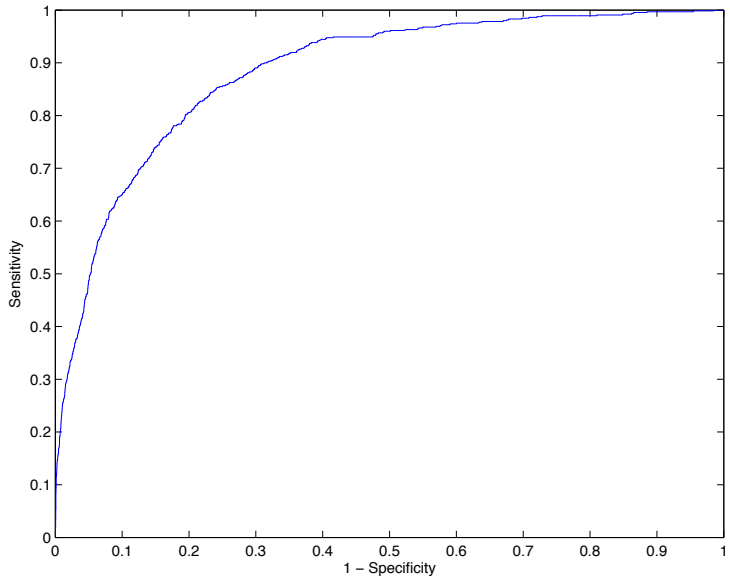


Figure 5.1: ROC curve of logistic model's predictive performance under retrospective analysis.

Figure 5.1 plots the ROC curve of the model's performance under a range of thresholds. The X axis plots 1 - the specificity for a given threshold. The Y axis plots the corresponding sensitivity at the same threshold. Both performance metrics are important from a clinical perspective. Higher X values correspond to a larger number of false alarms. Higher Y values correspond to more patients correctly being identified for early intervention.

Based on these results, a threshold of $y = 0.9323$ was chosen to achieve a specificity close to 95%. This specificity value was chosen in turn to generate only a manageable number of alerts per hospital floor per day. Even at this relatively high specificity, the logistic regression approach achieves a sensitivity of 48.8%. Table 5.2 summarizes the performance at this cutpoint under several common statistical metrics.

Area under curve	0.8834
Specificity	0.9500
Sensitivity	0.4877
Positive predictive value	0.3138
Negative predictive value	0.9753
Accuracy	0.9292

Table 5.2: The predictive performance of logistic regression at a chosen cutpoint under retrospective analysis

Real-Time Simulation

In order to train the logistic model, our retrospective analysis looked at only a single 24-hour window of data for each patient. In a system that predicts patients' outcomes in real time, their scores would be recomputed each time new data is entered into the EMR database. Hence, each patient will effectively have a series of scores over the length of their hospital stay, and an alarm will be triggered when *any one* of these scores is above the threshold.

To better understand the model's performance under these circumstances, we developed a real-time simulator which "replays" each patient's EMR data in chronological order over their entire hospital stay. For each entry in the EMR dataset, the simulator collects all the patient's data over a 24-hour window ending at the new entry, and computes a new score for the patient (using the β values fitted during the retrospective study). ICU admission is predicted if any one of the patient's scores exceed a given threshold.

For this real-time simulation, we replayed a smaller, more recent data set cataloging 1,284 hospital visits from 1,204 distinct patients between October 2010 and December 2010. Figure 5.2 plots the corresponding ROC curve of the model's performance.

Area under curve	0.7293
Specificity	0.9492
Sensitivity	0.4127
Positive predictive value	0.2955
Negative predictive value	0.9691
Accuracy	0.9229

Table 5.3: The predictive performance of logistic regression at a chosen cutpoint under real-time simulation

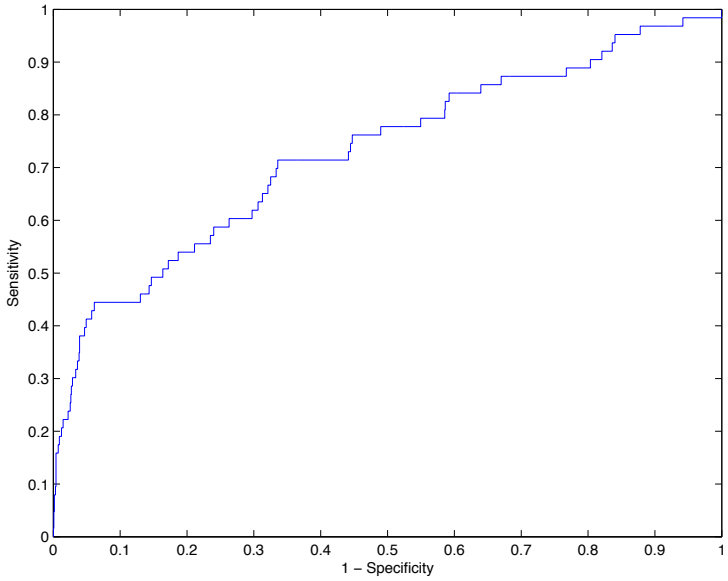


Figure 5.2: ROC curve of logistic model's predictive performance under real-time simulation.

As with the retrospective study, we chose a threshold to achieve a specificity close to 95%, yielding a cutpoint of $y = 0.9987$. At a specificity of 94.9%, the model achieves a sensitivity of 41.3%. Table 5.3 summarizes the model's performance at this cutpoint.

5.3 Real-Time Data Collection System

Real-time data collection forms a key component of the two-tiered clinical early warning system described in this chapter. The long-term goal for this component is to collect vital sign data from patients in real time, and combine this real-time data with existing electronic medical records to better predict the patient's outcome.

In this section, we discuss an ongoing clinical trial of a WSN-based monitoring system. The trial, deployed on four wards on three floors of Barnes-Jewish Hospital, uses WSN-based devices to collect pulse and blood-oxygenation data from patients in real time. The data collected is *not* yet being analyzed in real time, but rather will form a dataset for designing such a real-time clinical event detection component. This section will focus on important new lessons learned while deploying a trial of this scale, to serve as guidelines for future clinical trials based on WSN technology.

5.3.1 Background

The hardware and software system used in this clinical trial are based on an earlier pilot study performed in a step-down unit at Barnes-Jewish Hospital [20]. Enrolled patients are provided devices which periodically collect pulse and blood oxygenation level data, and route the data to a wired access point using an onboard radio based on the IEEE 802.15.4 radio standard. WSN network coverage is achieved by plugging additional TelosB nodes into electrical outlets in patients' rooms and in the hallway. These relay nodes form a dedicated routing infrastructure, which patient nodes locate using the *Dynamic Relay Association Protocol* (DRAP) protocol [19].

The Collection Tree Protocol (CTP) [34] is employed on relay nodes to establish a data collection tree rooted at a wired base station. Once data reaches a base station, it is logged in a local PostgreSQL database. In addition to patient data, relay nodes send "beacon" packets to the closest base station at a rate of one packet per minute. These beacons provide diagnostic information about the network's operation even when no patients are enrolled in the study.

There are two important differences between the earlier pilot study and the ongoing clinical trial described in this section. First, the pilot study was generally isolated from the BJC network. Vital sign data was kept locally on the base station; the building's wireless network was used only for remote administration of the base station. Under the clinical trial, the base stations use BJC's internal network to connect to a central EMR database. Second, the clinical trial operates at a much larger scale than the pilot study. The pilot study deployed 18 relay nodes and one base station in one step-down unit. The clinical trial was initially deployed with 30 relay nodes and two base stations across two wards, and was expanded to 59 relay nodes and four base stations across three floors. Although these differences are subtle, they have both led to significant deployment challenges discussed below.

5.3.2 Deployment Challenges

The increased scale of the clinical trial, and its interaction with external database systems, have had significant effects on its reliability. In this subsection, we discuss some of the unique challenges faced during the first months of the clinical trial. We discuss preliminary solutions to these challenges in the following subsection.

Sensing reliability. Sensing reliability posed a significant challenge during the earlier pilot study, with an average reliability of 80.85% (compared to a network reliability of 99.68%). The pilot study recommended combating the low sensing reliability by sending a "disconnection alarm" if long-lived sensor failures are encountered. This recommendation was implemented for the clinical trial. Nevertheless, sensor reliability was even worse at the beginning of the clinical trial: out of 9 patients, sensor reliability had a mean of 33.84% and median value of 27.95%.

Equipment disconnection. Over the first 3 months of the deployment, 12 relay nodes (out of the initial 30 deployed) disappeared from the network. Several of these nodes were found physically unplugged from the wall, while the others have still not been located. 11 of these disconnections occurred in a short period of time, taking down a critical part of the WSN infrastructure. During this time, one of the two base stations' 802.15.4 interface was also found unplugged, sitting on top of the laptop's lid; without the interface, the base station could no longer collect data from the WSN. Combined, these disconnections effectively cut off most of the relay network from both base stations.

Unpowered patient devices. Study nurses must install batteries into the patient device when a new patient is enrolled. However, the nurses were not initially given clear instructions on how to install batteries, and the patient devices gave no obvious feedback that they were powered on. As a result, patients enrolled during the study's first two weeks were provided devices which did not power up, resulting in total data loss until the problem was identified.

Unexpected software interactions. Three of the four base stations were provided by BJC, and were prepared with a standard Windows XP laptop image. This image included a backup utility configured to automatically run backups two nights a week. During the clinical trial, it was discovered that this utility would lock files while backing them up, causing the base stations' PostgreSQL server instances to crash. Hence, the first weeks of the deployment were plagued with unexplained base station crashes until the backup schedule was discovered and disabled.

Moreover, the standard laptop image includes remote administration software that allows IT to push out critical security updates, in turn requiring monthly reboots. The data collection software ran as an ordinary user-level application and was not designed to automatically restart after a reboot, resulting in data loss until the reboots were discovered and the software was manually restarted. The rebooting issue was

compounded by the full-disk encryption included in the image. After rebooting, the base stations prompted for a passphrase to unlock the Windows partition, requiring physical access to the machine.

5.3.3 Lessons Learned

Education and training. Many of the early reliability problems with the study stemmed from insufficient education about the equipment used in the study. We initially treated sensor reliability as an unavoidable problem which could only be combated through oversampling and disconnection alarms. When both techniques failed to improve reliability, a team member persistently contacted the pulse-ox sensor's vendor; he eventually received specific recommendations on sensor placement and maintenance, which were not all intuitive from the sensor labeling. After training the study nurse with the vendor's recommendations, sensing reliability has improved dramatically.

Better training of the study nurses would also have prevented the first patient devices being deployed without properly-installed batteries.

Equipment labeling. The relay devices and base station were initially deployed without any labels indicating that they were part of a clinical monitoring system. Although most of the disconnected devices have not yet been found, it seems likely that some were unplugged and set aside out of curiosity or suspicion; we fielded many questions about the devices from curious nurses, doctors, and cleaning staff passing by the deployment, including a nurse who asked if a relay device was a bomb. We also believe that some devices may have been unplugged by cleaning staff who needed the power outlets, since there was no indication that the device was important.

New relay devices are now deployed with laminated labels, with clear instructions that they should not be unplugged (and should be returned to the nurses' station if found disconnected). The label is also designed to conceal the relay nodes' exposed circuit board, making them appear less suspicious. No new disconnections have occurred since adding the labels.

Equipment feedback. After discovering the battery installation issue, patient devices were programmed with a new firmware that lit the sensor's LED for 30 seconds after being turned on. This change provided visual feedback to the study nurse that

the device was functioning, eliminating the possibility of more patients being given powered-down devices.

Infrastructure monitoring. When the system was initially deployed, there was no automated process in place to ensure that the base stations were still running. Hence, base station downtimes often went undiscovered for many days, and were only noticed when new patients were enrolled. Since the initial deployment, we have modified the base station software to E-mail nightly diagnostic reports, providing feedback that they are still operational.

For similar reasons, relay node disconnections often went unnoticed until they began to affect network performance. Labeling the equipment has so far prevented the problem from recurring. In case the problem recurs, we are exploring adding backup battery power to the relay nodes, and sending alerts to the base station when they switch over to battery power.

Base station design. Many of the reliability issues at the base station stemmed from the base station software being built as an ordinary user-level application on top of a standard Windows XP image. The XP image was designed for laptops which may hold confidential data and are prone to theft and data loss, and where routine reboots are acceptable. In retrospect, deploying the base station software on top of a minimal Linux or Windows Server installation would have eliminated these problems from the outset.

We have since worked around many of these issues by removing the backup and full-disk encryption software, and modifying the base station software to run as a system-level service. While this solution does not eliminate the monthly reboots, it eliminates the manual intervention needed to bring the base stations back up after a reboot.

Chapter 6

Conclusions

This dissertation discussed four different efforts in designing energy-efficient WSNs. ART, discussed in Chapter 2, is a representative approach targeted at a specific part of the hardware stack (specifically, the radio subsystem). ART reduces the energy consumed by packet transmissions by an average of 15% with no loss in link quality, and up to 40% under heavy contention. Moreover, ART is largely agnostic to the routing and application layers which sit on top of it, making it broadly applicable to a wide range of applications. Chapters 3 and 4 cover two distributed WSN architectures designed specifically to improve the lifetime of structural health monitoring applications. In contrast to ART, the techniques in these chapters are targeted specifically to a particular application (decentralized SHM) but offer even greater opportunity for energy savings. By leveraging the motes' onboard computational power, data transmissions are reduced by up to 99.8% with no loss in damage identification or localization performance.

Finally, Chapter 5 described my work toward a real-time clinical monitoring and event detection system. The logistic regression study serves as a proof-of-concept for clinical event detection, while also identifying candidate patients for an ongoing clinical trial. More generally, the lessons learned during the clinical trial may also guide future trials of clinical WSNs.

Looking ahead, the lessons learned in applying holistic codesign to SHM may also become an important part of future real-time clinical monitoring systems. At an architectural level, the monitoring system currently under trial at BJH resembles a centralized SHM system. After patient devices receive a valid reading from the sensor, this data is relayed to the base station and processed in a centralized fashion. This simple approach works well for a pulse-ox sensor, where we collect only one reading

(consisting of a few bytes of data) per minute and hence transmission costs are not a major issue.

However, other kinds of sensors or sensing modes may require higher data rates which are impractical to support in a centralized system. For example, ECG sensors may be used to noninvasively estimate heart-rate variability (HRV), which may act as an early indicator of serious medical conditions [12, 35, 85]. Measuring HRV requires sampling the ECG continuously over a long time window, which would be impractically expensive to send back to a base station. Alternatively, a holistically codesigned system could exploit the patient devices' onboard processing capabilities to analyze HRV onboard, and then only relay data to the base station when the HRV is abnormal. Similar approaches have been highly effective at reducing data transmission costs in SHM applications, making a compelling argument for its use in future clinical monitoring systems as well.

References

[1] http://www.tinyos.net/.

[2] http://www.cse.wustl.edu/wsn.

[3] http://shm.cs.uiuc.edu/software.html.

[4] http://wsn.cse.wustl.edu/index.php/MLA.

[5] American Society of Civil Engineering. 2009 report card for America's infrastructure. http://www.asce.org/reportcard/2009/.

[6] David Arney, Miroslav Pajic, Julian M. Goldman, Insup Lee, Rahul Mangharam, and Oleg Sokolsky. Toward patient safety in closed-loop medical device systems. In *ICCPS*, 2010.

[7] R. Bailon, L. Sornmo, and P. Laguna. A robust method for ECG-based estimation of the respiratory frequency during stress testing. *IEEE Transactions on Biomedical Engineering*, 53(7):1273 –1285, July 2006.

[8] Dionisio Bernal and Burcu Gunes. Flexibility based approach for damage characterization: Benchmark application. *Journal of Engineering Mechanics*, 130(1):61–70, January 2004.

[9] Douglas M. Blough, Mauro Leoncini, Giovanni Resta, and Paolo Santi. The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(9), 2006.

[10] Douglas M. Blough, Mauro Leoncini, Giovanni Resta, and Paolo Santi. Topology control with better radio models: Implications for energy and multi-hop interference. *Perform. Eval.*, 64(5), 2007.

[11] 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. *New England Journal of Medicine*, 324(6):370–6, Feb 1991.

[12] Timothy Buchman, Phyllis Stein, and Brahm Goldstein. Heart rate variability in critical illness and critical care. *Current Opinion in Critical Care*, 8(4):311–315, 2002.

[13] 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. *Medical Journal of Australia*, 171(1):22–5, Jul 1999.

[14] Martin Burkhart, Pascal von Rickenbach, Roger Wattenhofer, and Aaron Zollinger. Does topology control reduce interference? In *MobiHoc*, 2004.

[15] N. Castaneda, F. Sun, S. Dyke, C. Lu, A. Hope, and T. Nagayama. Implementation of a correlation-based decentralized damage detection method using wireless sensors. In *ASEM Conference*, 2008.

[16] Matteo Ceriotti, Luca Mottola, Gian Pietro Picco, Amy L. Murphy, Stefan Guna, Michele Corra, Matteo Pozzi, Daniele Zonta, and Paolo Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *IPSN*, 2009.

[17] Alberto Cerpa, Jennifer L. Wong, Miodrag Potkonjak, and Deborah Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *MobiHoc*, 2005.

[18] Kameswari Chebrolu, Bhaskaran Raman, Nilesh Mishra, Phani Kumar Valiveti, and Raj Kumar. BriMon: a sensor network system for railway bridge monitoring. In *MobiSys*, 2008.

[19] Octav Chipara, Christopher Brooks, Sangeeta Bhattacharya, Chenyang Lu, Roger D. Chamberlain, Gruia-Catalin Roman, and Thomas C. Bailey. Reliable real-time clinical monitoring using sensor network technology. In *AMIA*, 2009.

[20] Octav Chipara, Chenyang Lu, Thomas C. Bailey, and Gruia-Catalin Roman. Reliable clinical monitoring using wireless sensor networks: Experience in a step-down hospital unit. In *SenSys*, 2010.

[21] E.H. Clayton. Development of an experimental model for the study of infrastructure preservation. In *National Conference on Undergraduate Research*, 2002.

[22] E.H. Clayton. Frequency correlation-based structural health monitoring with smart wireless sensors. Master's thesis, Washington University in St. Louis, 2006.

[23] Crossbow Technology. MICA2 wireless measurement system. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[24] Crossbow Technology. MICAz wireless measurement system. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[25] Crossbow Technology, Inc. *ITS400 Imote2 Basic Sensor Board*.

[26] Crossbow Technology, Inc. *Imote2 Hardware Reference Manual*, 2007.

[27] A. Culotta, D. Kulp, and A. McCallum. Gene prediction with conditional random fields. *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2005-028*, 2005.

[28] Zhongdong Duan, Guirong Yan, and Jinping Ou. Structural damage detection using the angle-between-string-and-horizon flexibility (under review). *Structural Engineering and Mechanics*.

[29] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. Four-bit wireless link estimation. In *HotNets VI*, 2007.

[30] 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.

[31] M.V. Gangone, M.J. Whelan, K.D. Janoyan, K. Cross, and R. Jha. Performance monitoring of a bridge superstructure using a dense wireless sensor network. In *International Workshop on Structural Health Monitoring, Stanford, California*, 2007.

[32] 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.

[33] Yan Gao, Jennifer C. Hou, and Hoang Nguyen. Topology control for maintaining network connectivity and maximizing network capacity under the physical model. In *INFOCOM*, 2008.

[34] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *SenSys*, 2009.

[35] M. Pamela Griffin and J. Randall Moorman. Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. *Pediatrics*, 107(1):97–104, 1 2001.

[36] X. He, R.S. Zemel, and M.A. Carreira-Perpinán. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition*, 2004.

[37] D.W. Hosmer and S. Lemeshow. *Applied logistic regression*. Wiley-Interscience, 2000.

[38] 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.

[39] A. E. Jones, M. D. Brown, S. Trzeciak, N. I. Shapiro, J. S. Garrett, A. C. Heffner, and J. A. Kline. The effect of a quantitative resuscitation strategy on mortality in patients with sepsis: a meta-analysis. *Crit. Care Med.*, 36:2734–2739, Oct 2008.

[40] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[41] 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. *AMIA*, 2007.

[42] 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 *IPSN*, 2007.

[43] Kevin Klues, Gregory Hackmann, Octav Chipara, and Chenyang Lu. A component based architecture for power-efficient media access control in wireless sensor networks. In *SenSys*, 2007.

[44] Kevin Klues, Vlado Handziski, Chenyang Lu, Adam Wolisz, David Culler, David Gay, and Philip Levis. Integrating concurrency control and energy management in device drivers. In *SOSP*, 2007.

[45] W A Knaus, E A Draper, D P Wagner, and J E Zimmerman. Apache ii: a severity of disease classification system. *Crit Care Med*, 13(10):818–829, Oct 1985.

[46] Jeonggil Ko, Jong Hyun Lim, Yin Chen, Răzvan Musvaloiu-E, Andreas Terzis, Gerald M. Masson, Tia Gao, Walt Destler, Leo Selavo, and Richard P. Dutton. Medisn: Medical emergency detection in sensor networks. *ACM Trans. Embed. Comput. Syst.*, 10:11:1–11:29, August 2010.

[47] B. H. Koh and S. J. Dyke. Structural damage detection in cable-stayed bridges using correlation and sensitivity of modal data. *Computers and Structures*, 85:117—130, 2007.

[48] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[49] Dhananjay Lal, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *IEEE GLOBECOM*, 2003.

[50] E. C. Levy. Complex-curve fitting. *IRE Transactions on Automatic Control*, 4:37—44, 1959.

[51] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, and Roger Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *ACM PODC*, 2001.

[52] Ning Li, J. C. Hou, and L. Sha. Design and analysis of an MST-based topology control algorithm. *IEEE Transactions on Wireless Communications*, 4(3):1195–1206, 2005.

[53] Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, John A. Stankovic, and Tian He. ATPC: adaptive transmission power control for wireless sensor networks. In *SenSys*, 2006.

[54] K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[55] J.P. Lynch and K. Loh. A summary review of wireless sensors and sensor networks for structural health monitoring. *Shock and Vibration Digest*, 38(2):91–128, 2006.

[56] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA*, 2002.

[57] David Malan, Thaddeus Fulford-jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[58] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM.

[59] A. Messina, I. A. Jones, and E. J. Williams. Damage detection and localization using natural frequency changes. In *Conference on Identification in Engineering Systems, U.K.*, 1996.

[60] A. Messina, E. J. Williams, and T. Contursi. Structural damage detection by a sensitivity and statistical-based method. *Journal of Sound and Vibration*, 215(5):791–808, 1998.

[61] David Moss and Philip Levis. BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Technical Report SING-08-00, Stanford Information Networks Group, 2008.

[62] T. Nagayama. *Structural Health Monitoring Using Smart Sensors*. PhD thesis, University of Illinois at Urbana-Champaign, 2007.

[63] C. Orsenigo and C. Vercellis. Time series classification by discrete support vector machines. In *Artifical Intelligence and Data Mining Workshop*, 2006.

[64] Shamim N. Pakzad, Gregory L. Fenves, Sukun Kim, and David E. Culler. Design and implementation of scalable wireless sensor network for structural monitoring. *ASCE Journal of Infrastructure Engineering*, 14(1):89–101, March 2008.

[65] A.K. Pandey and M. Biswas. Experimental verification of flexibility difference method for locating damage in structures. *Journal of Sound and Vibration*, 184(2):311–328, 1995.

[66] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *IPSN*, 2005.

[67] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *IEEE MASS*, 2004.

[68] Jennifer A. Rice and B. F. Spencer, Jr. Structural health monitoring sensor development for the Imote2 platform. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, 2008.

[69] J. Rodríguez, C. Alonso, and H. Boström. Learning first order logic time series classifiers: Rules and boosting. *Principles of Data Mining and Knowledge Discovery*, pages 85–105, 2000.

[70] Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, 2005.

[71] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

[72] Dongjin Son, B. Krishnamachari, and J. Heidemann. Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks. In *IEEE SECON*, 2004.

[73] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. Experimental study of transmission power control and blacklisting based link quality control in wireless sensor networks. Technical Report ISI-TR-629, Information Sciences Institute, 2007.

[74] Wei Song, Shirley Dyke, GunJin Yun, and Thomas Harmon. Improved damage localization and quantification using subset selection. *Journal of Engineering Mechanics*, 135(6):548–560, 2009.

[75] B.F. Spencer and T. Nagayama. Smart sensor technology: a new paradigm for structural health monitoring. In *Asia-Pacific Workshop on Structural health Monitoring, Yokohama, Japan*, 2006.

[76] Kannan Srinivasan and Philip Levis. RSSI is under appreciated. In *EmNets*, 2006.

[77] STMicroelectronics. *LIS3L02DQ MEMS Inertial Sensor*, 2005.

[78] Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.

[79] Texas Instruments. *CC1000 Single Chip Very Low Power RF Transceiver*.

[80] The Joint Commission. 2008 national patient safety goals.

[81] S. W. Thiel, J. M. Rosini, W. Shannon, J. A. Doherty, S. T. Micek, and M. H. Kollef. Early prediction of septic shock in hospitalized patients. *J Hosp Med*, 5:19–25, Jan 2010.

[82] S.B. Wang, A. Quattoni, L.P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1521–1527, 2006.

[83] R. Wattenhofer and A. Zollinger. XTC: a practical topology control algorithm for ad-hoc networks. In *IPDPS*, 2004.

[84] Jian-Huang Weng, Chin-Hsiung Loh, Jerome P. Lynch, Kung-Chun Lu, Pei-Yang Lin, and Yang Wang. Output-only modal identification of a cable-stayed bridge using wireless monitoring systems. *Engineering Structures*, 30(7):1820–1830, 2008.

[85] Karl Werdan, Hendrik Schmidt, Henning Ebelt, Klaus Zorn-Pauly, Bernd Koidl, Robert Sebastian Hoke, Konstantin Heinroth, and Ursula Müller-Werdan. Impaired regulation of cardiac function in sepsis, SIRS, and MODS. *Canadian Journal of Physiology and Pharmacology*, 87(4):266–274, April 2009.

[86] 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. *Medical Journal Australia*, 163(9):458–71, 1995.

[87] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys*, 2003.

[88] 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, University of Virginia, Dec 2006.

[89] Guoliang Xing, Chenyang Lu, and Robert Pless. Localized and configurable topology control in lossy wireless sensor networks. In *ICCCN*, 2007.

[90] Ning Xu, Sumit Rangwala, Krishna Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *SenSys*, 2004.

[91] Guirong Yan, Zhongdong Duan, and Jinping Ou. An axial strain flexibility for damage detection of truss structure. In *International Workshop on Smart Materials and Structures*, 2005.

[92] Pedro Pablo Espana Yandiola, Alberto Capelastegui, Jose Quintana, Rosa Diez, Inmaculada Gorordo, Amaia Bilbao, Rafael Zalacain, Rosario Menendez, and Antonio Torres. Prospective comparison of severity scores for predicting clinically relevant outcomes for patients hospitalized with community-acquired pneumonia. *Chest*, 135(6):1572–1579, Jun 2009.

[93] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *SenSys*, 2006.

[94] Gun Jin Yun, Kenneth A. Ogorzalek, Shirley J. Dyke, and Wei Song. A two-stage damage detection approach based on subset selection and genetic algorithms. *Smart Structures and Systems*, 5:1–21, 2009.

[95] B Zernikow, K Holtmannspoetter, E Michel, W Pielemeier, F Hornschuh, A Westermann, and K H Hennecke. Artificial neural network for risk assessment in preterm neonates. *Archives of Disease in Childhood - Fetal and Neonatal Edition*, 79(2):F129–F134, 1998.

[96] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *SenSys*, 2004.

[97] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys*, 2003.

[98] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys*, 2004.

[99] Andrew T. Zimmerman, Michihito Shiraishi, R. Andrew Swartz, and Jerome P. Lynch. Automated modal parameter estimation by parallel processing within wireless monitoring systems. *Journal of Infrastructure Systems*, 14(1):102–113, March 2008.

[100] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *IEEE SECON*, 2004.

Vita

Gregory W. Hackmann

Date of Birth December 13, 1981

Place of Birth St. Louis, Missouri

Degrees Ph.D., Computer Science, Washington University in St. Louis, August 2011
B.S. Magna Cum Laude, Computer Science, Washington University in St. Louis, May 2004

Selected Publications

Hackmann, G., Chen, M., Chipara, O., Lu, C., Chen, Y., Kollef, M., and Bailey, T.C., "Toward a Two-Tier Clinical Warning System for Hospitalized Patients", *AMIA*, 2011. (to appear)

Hackmann, G., Guo, W., Yan, G., Lu, C., and Dyke, S., "Cyber-Physical Codesign of Distributed Structural Health Monitoring With Wireless Sensor Networks," *ACM/IEEE IC-CPS*, 2010.

Hackmann, G., Fei Sun, Castaneda, N., Lu, C., and Dyke, S., "A Holistic Approach to Decentralized Structural Damage Localization Using Wireless Sensor Networks," *IEEE RTSS*, 2008.

Hackmann, G., Chipara, O., and Lu, C., "Robust Topology Control for Indoor Wireless Sensor Networks," *ACM SenSys*, 2008.

August 2011