NSEL Report Series
Report No. NSEL-001
November 2007

Structural Health Monitoring Using Smart Sensors

Tomonori Nagayama
and
Billie F. Spencer, Jr.

NEWMARK STRUCTURAL ENGINEERING LABORATORY

Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign

UILU-ENG-2007-1801

ISSN: 1940-9826

Contact:
Prof. B.F. Spencer, Jr.
Director, Newmark Structural Engineering Laboratory
2213 NCEL, MC-250
205 North Mathews Ave.
Urbana, IL 61801
Telephone (217) 333-8630
E-mail: bfs@uiuc.edu

This technical report is based on the first author's doctoral dissertation under the same title which was completed in April 2007. The second author served as the dissertation advisor for this work.

Financial support for this research was provided in part by the National Science Foundation (NSF) under NSF grants CMS 03-01140 and CMS 06-00433 (Dr. S. C. Liu, Program Manager). The first author was supported by a Vodafone-U.S. Foundation Graduate Fellowship. This support is gratefully acknowledged.

The cover photographs are used with permission. The Trans-Alaska Pipeline photograph was provided by Terra Galleria Photography (http://www.terragalleria.com/).

ABSTRACT

Industrialized nations have a huge investment in the pervasive civil infrastructure on which our lives rely. To properly manage this infrastructure, its condition or serviceability should be reliably assessed. For condition or serviceability assessment, Structural Health Monitoring (SHM) has been considered to provide information on the current state of structures by measuring structural vibration responses and other physical phenomena and conditions. Civil infrastructure is typically large-scale, exhibiting a wide variety of complex behavior; estimation of a structure's state is a challenging task. While SHM has been and still is intensively researched, further efforts are required to provide efficient and effective management of civil infrastructure.

Smart sensors, with their on-board computational and communication capabilities, offer new opportunities for SHM. Without the need for power or communication cables, installation cost can be brought down drastically. Smart sensors will help to make monitoring of structures with a dense array of sensors economically practical. Densely installed smart sensors are expected to be rich information sources for SHM.

Efforts toward realization of SHM systems using smart sensors, however, have not resulted in full-fledged applications. All efforts to date have encountered difficulties originating from limited resources on smart sensors (e.g., small memory size, small communication throughput, limited speed of the CPU, etc.). To realize an SHM system employing smart sensors, this system needs to be designed considering both the characteristics of the smart sensor and the structures to be monitored.

This research addresses issues in smart sensor usages in SHM applications and realizes, for the first time, a scalable and extensible SHM system using smart sensors. The architecture of the proposed SHM is first presented. The Intel Imote2 equipped with an accelerometer sensor board is chosen as the smart sensor platform to demonstrate the efficacy of this architecture. Middleware services such as model-based data aggregation, reliable communication, and synchronized sensing are developed. SHM Algorithms identified as promising for the usage on smart sensor systems are extended to improve practicability and implemented on Imote2s. Careful attention has been paid to integrating these software components so that the system possesses identified desirable features.

The damage detection capability and autonomous operation of the developed system are then experimentally verified. The SHM system consisting of ten Imote2s are installed on a scale-model truss. The SHM system monitors the truss in a distributed manner to localize simulated damage.

In summary, this report proposes and realizes a scalable and autonomous SHM system using smart sensors. The system is experimentally verified to be effective for damage detection. The autonomous nature of the system is also demonstrated. Successful completion of this research paves the way toward full-fledged SHM systems employing a dense array of smart sensors. The software developed under this research effort is open-source and is available at: http://shm.cs.uiuc.edu/.

Contents

Page

Chapter 1

INTRODUCTION

1.1 Monitoring of civil infrastructure

Our lives rely heavily on the pervasive civil infrastructure in which industrialized nations have huge investments. Malfunctioning of civil infrastructure has caused tremendous economic loss and claimed numerous human lives. Civil infrastructure is, thus, critical to keep our economy running, while the infrastructure itself is an important asset to be managed.

To properly manage civil infrastructure, its condition, or serviceability, must be assessed. Many variables can be monitored and used for the assessment. For instance, Intelligent Transportation Systems make use of traffic surveillance information to efficiently manage the transportation system. Tunnels are monitored for traffic accidents and air quality. The Urgent Earthquake Detection and Alarm System (Nakamura, 2004) detects primary seismic waves and stops trains before severe secondary waves approach. Measurement and proper data processing are expected to give a reasonable assessment of serviceability that can then be improved based on the assessment.

The physical state of a structural system, for example, applied load, vibration level, and existence of structural damage, is among the factors that determine serviceability. Sensing physical quantities in detail offers the potential to better estimate structural conditions. For river bank protection, for instance, water level may be monitored and the associated load estimated. Precipitation rate and groundwater level are important indicators to predict slope failure. Strain and temperature measurements can be utilized to monitor concrete gravity or arch dams. Engineers, owners, and users can make better decisions based on the measured information.

Structural condition assessment is, however, not always straightforward as in the case of the Structural Health Monitoring (SHM) of buildings, bridges, and towers. The structural condition is oftentimes sought in terms of structural characteristics, i.e., mass, damping, stiffness matrices, damage existence, and/or applied load to the system. These structures are large and consist of many members, which makes such structural condition assessment difficult and/or prohibitively expensive. One approach in SHM to alleviate this difficulty is based on vibration measurement. Though structural characteristics and applied load are difficult to assess directly, dynamic behavior, which is a function of the structural characteristics and applied load, can be measured. The structural characteristics and applied load information lurk in the dynamic behavior. Structural soundness is expected to be estimated by inverse analyses of the dynamic behavior.

Because buildings, bridges, and towers are typically large and complex, information from just a few sensors is inadequate to accurately assess the structural condition. The dynamic behavior of these structures is complex in both spatial and time scale. Moreover, damage/deterioration is intrinsically a local phenomenon. Therefore, to comprehend such dynamic behavior, the motion of structures needs to be monitored by densely located sensors at a sampling frequency sufficiently high to capture salient dynamic characteristics.

1.2 SHM using smart sensors

When many sensors are implemented, wireless communication appears to be attractive. The high cost associated with the installation of wired sensors (Celebi, 2002; Farrar, 2001) can be greatly reduced by employing wireless sensors. Wireless sensors often convert analog signals to digital signals prior to radio frequency (RF) transmission, while many wired systems collect analog signals at one or several base stations where the signals conversion takes place. The digital conversion on the wireless sensor node eliminates possible signal degradation during analog signal communication through long cables. Wireless sensor systems are, thus, promising as data acquisition systems with a large number of sensors installed on sizable structures.

Being "smart", i.e., having data processing capability in the sensors, is an essential feature that further increases the potential of wireless sensors. Smart sensors can locally process measured data and transmit only the important information through wireless communication. As a network, wireless sensors extend the capability. For instance, sensors that are malfunctioning in the network can be detected, and other sensors can rebuild sensor topology without this dead node. As another instance, location mapping can be done automatically by a localization service (Doherty et al., 2001; Kwon et al., 2005a; Kwon et al., 2005b), which helps civil engineers determine and confirm the location of large numbers of sensors on complex structures.

Smart sensors, however, have limited resources, prohibiting direct application of traditional structural monitoring strategies. For example, the communication speed is too slow to centrally collect all of the measured information. Clocks on sensor nodes are not always synchronized. Some communication packets may be lost. Storage and memory space is limited. Processor speed is slower than that of a PC. Smart sensors do not necessarily offer a real-time system; programmers may not be able to assign appropriate priority to given tasks. Moreover, battery power imposes limitations on many aspects of smart sensors. Any task consuming large amounts of power becomes impractical on a battery-operated smart sensor node. Smart sensor systems need to overcome these limitations using deliberate system design, as seen in some of the time synchronization and reliable communication research efforts (Ganeriwal et al., 2003; Maroti et al. 2004; Mechitov et al., 2004).

From the perspective of SHM, being smart makes it feasible to monitor structural response densely both in time and space. The amount of data generated from a monitored structure can be enormous due to the large number of sensors and high sampling frequency. For example, the Tsing Ma and Kap Shui Mun Bridges in Hong Kong produce 63 MB of data every hour (Wong, 2004). Being smart is expected to allow significant data compression at the node level by extracting only the information necessary for the task at hand, thus reducing the amount of data to be stored or transferred through wireless communication.

Furthermore, being smart gives the possibility of autonomous structural health monitoring, with reduced user interaction. Smart sensors communicate with each other through the RF link to share measurement data. The data can be utilized to judge structural soundness. Once the smart sensor network detects structural damage, the network informs users about the damage or necessary repair. Microprocessors on the smart sensors make it possible to perform this procedure autonomously.

Many smart sensor prototypes have been developed and several attempts to use smart sensors for SHM are reported so far. There are, however, many problems to be solved. Ruiz-Sandoval (2004) addressed some sensor hardware problems from a civil engineering viewpoint. As for algorithms, most attempts at SHM with smart sensors only substitute the wired link with wireless communication and apply traditional damage detection algorithms at the base station. An SHM system with such algorithms assuming central data collection does not scale to a large number of smart sensors. Researchers have also proposed approaches where simple data processing is performed on each smart sensor node without interaction with other nodes; these attempts do not employ spatial information, and, therefore, have room for improvement in terms of damage detection capability. Information from a dense array of smart sensors should be processed in a coordinated manner, rather than independently. SHM algorithms for distributed and coordinated data processing making use of the smart sensor's distributed computing and sensing resources have only recently appeared.

Gao (2005) recently proposed a new distributed computing strategy (DCS) for SHM envisioning smart sensor usage. DCS is intended to use the smart sensor's data processing capability in a coordinated way to achieve SHM. Computer analysis and experimental validation on a simulated wireless network showed DCS for SHM is promising. Implementation of DCS on smart sensors by addressing implementation issues and experimental validation of the proposed scheme are imperative.

1.3 Overview of report

This research focuses on the realization of a vibration-based SHM framework employing smart sensors that can detect and localize damage. In general terms, damage can be defined as changes introduced into a system that adversely affect the current or future performance of that system (Doebling et al., 1998). The effect of damage on a structure can be classified as linear or nonlinear. A linear damage situation is defined as the case when the structure remains linear-elastic after damage, while damage introducing nonlinear behavior is defined as nonlinear damage (Doebling et al., 1996). The type of damage considered in this research is linear damage. Metal corrosion, concrete spalling/ scour, and yielding of beam-column joints are typical linear damage examples of interest to civil engineers. Linear damage changes structural characteristics such as mass, damping, and stiffness. Subsequent changes in modal parameters are identified and utilized in damage detection. A framework for such a vibration-based SHM employing smart sensors is proposed herein. Among the important features to be obtained are scalability to a large number of smart sensors and autonomous operation, as well as effective damage detection capability.

Chapter 2 provides the background of this research. Smart sensors, middleware services, and SHM are briefly reviewed. Research efforts employing smart sensors for SHM applications are then summarized and difficulties in these attempts are addressed. In the subsequent chapters, an SHM framework is realized on a smart sensor network that resolves the majority of these difficulties.

Chapter 3 describes the SHM architecture developed in this research. A homogeneous hardware configuration is selected, while smart sensor nodes are functionally differentiated into several categories. Smart sensors, middleware services, and damage detection algorithms used in such networks are briefly explained.

In Chapter 4, sensor boards for one of the smart sensor platforms, the Mica2, are developed to demonstrate sensor board customizability according to SHM requirements. Because strain sensors for the Mica2 were not available, a strain sensor board is developed as well as an Anti-Aliasing (AA) filter board. Scale-model experiments show that these sensor boards can facilitate accurate measurement of structural responses.

Middleware services realized as part of this research for SHM applications are discussed in Chapter 5. Middleware services include reliable communication, model-based data aggregation, and synchronized sensing. These middleware services can be used in a wide variety of civil engineering applications.

Chapter 6 discusses algorithms to be implemented on smart sensors. The DCS algorithm has the potential to realize densely deployed smart sensor networks for SHM because of its distributed and coordinated data processing. Algorithmic components of DCS are briefly reviewed. The damage detection algorithm in DCS is then extended by replacing the mass perturbation Damage Locating Vector (DLV) method with the Stochastic Damage Locating Vector (SDLV) method. The SDLV method is shown to simplify damage detection and reduce total power consumption. This chapter provides the algorithmic basis for the subsequent chapters.

In Chapter 7, the DCS algorithm is implemented on the Imote2 smart sensor platform using the middleware services and algorithms. First, numerical functions are ported to the Imote2. Second, the capabilities of the generic sensor board are examined. Third, each of the DCS algorithms is implemented on smart sensors, and its validity is numerically investigated.

Chapter 8 describes experimental verification of the developed framework. Smart sensor nodes are placed on a scale-model, three-dimensional truss. One of the bar elements of the truss is replaced with a more slender element to simulate linear damage to the truss. The smart sensor system measures acceleration responses of the model and localizes damage. Calculation and communication time, the battery life, and damage detection capability are discussed based on findings from the experiments.

Chapter 9 summarizes the research detailed in this report and presents possible directions for future research on SHM using smart sensors.

Chapter 2

BACKGROUND

This chapter presents the background for this research. Realization of an SHM framework using smart sensors requires interdisciplinary studies. The pursuit of a single component of the system does not necessarily realize the framework. Technical background and research efforts on three subjects closely associated with SHM using smart sensors, i.e., smart sensors, middleware services, and SHM algorithms, are overviewed. Research efforts directed toward SHM using smart sensors are then reviewed, and difficulties encountered in these attempts are summarized.

2.1 Smart sensor

2.1.1 Smart sensor's essential features

Smart sensor technology has been under rapid development in recent years. A smart sensor usually has five essential features: 1. on-board microprocessor, 2. sensing capability, 3. wireless communication, 4. battery-powered, and 5. low cost. This section describes each of these features in detail.

1. On-board microprocessor

The essential difference between a smart sensor and a standard integrated sensor is its intelligence, i.e., the on-board microprocessor. Programs can be embedded in the microprocessor, which allows smart sensors to save data locally, perform desired computations, make "if-then" decisions, scan necessary information, send results quickly, schedule multiple tasks, coordinate with surrounding sensors, etc. The on-board microprocessor can also control the time and duration that the sensor will be fully awake in order to efficiently manage power consumption. The smart sensors can arrange autonomous networks to achieve multiple tasks, such as SHM, power saving, multihop communication, self-configuration and self-healing of the network, dynamic routing, etc.

There are several options for embedding intelligence in a hardware design (Texas Instruments, 2007). Among the choices are: a digital signal processor (DSP), a Field Programmable Gate-Array (FPGA), an Application-Specific Integrated Circuit (ASIC), and a general purpose processor (GPP). Compared to the generally designed microprocessor, a DSP is specifically designed for rapid signal processing, such as real-time processing of audio signals on cell phones, often using an optimized instruction set (ISA). A multiply-accumulate (MAC) operation, which is suitable for matrix operation, such as convolution for filtering, dot product, or even polynomial evaluation, is commonly implemented on DSP chips. An FPGA has the capability of being reconfigurable within a system and offers greater raw performance per specific operation because of its dedicated logic circuit. However, FPGAs are more expensive and typically have higher power dissipation than DSPs and GPPs. ASICs can be tailored to perform specific functions extremely well, and can be made quite power efficient. However, since ASICS are not field-programmable, their functionality is not easily changed or updated. Also, without mass production, the initial setup costs are prohibitively high. Smart sensor applications require multiple tasks, such as wireless communication, data logging, data acquisition, and diverse signal processing. Among the above-mentioned options, GPPs are best suited for performing a broad array of tasks. When tasks on smart sensors include intensive data processing, a DSP or an FPGA can be used in combination with a GPP to improve data processing performance. Once smart sensor applications are well-understood, the necessary functions may be refined and ASIC may be implemented to perform specific tasks. Smart sensors on the market targeting a broad range of applications, in most cases, utilize only a GPP. If deemed advantageous, smart sensor systems developed on a GPP can then be improved by utilizing special purpose microprocessors such as DSP, FPGA, and ASIC.

2. Sensing capability

A smart sensor is able to convert the physical state of an object or environment such as temperature, light, sound, and/or motion into electrical or other types of signals that may be further processed. A single smart sensor node may have several sensors measuring different physical quantities. Micro-Electro-Mechanical System (MEMS) devices, which are the integration of mechanical elements, sensors, actuators, and electronics on a common silicon substrate through microfabrication technology, are often employed for sensors because of their small size, inexpensive cost (when mass produced), and low power consumption. Data acquisition parameters, such as sampling frequency and data length, can be controlled by the on-board processor. The on-board microprocessor can also access and process the acquired data. The sensing capability provides the interface between the smart sensor's on-board microprocessor and real-world physical phenomena.

One of the promising technologies in sensors is the use of quasi-digital sensors, such as frequency, duty-cycle, and pulse number output sensors, as proposed by Kirianaki et al. (2000) and Yurish (2005). Low noise sensitivity, wide dynamic range, and a simple interface which is directly connected to microprocessors are advantages of the quasi-digital sensors. The use of quasi-digital sensors is increasing. For example, the Intel Mote developed by Intel Corporation (Kling, 2004) utilizes a duty cycle output accelerometer.

3. Wireless communication

Smart sensors communicate with each other through a wireless link. While RF communication is most widely used, smart sensors with different transmission media, such as acoustic, laser, and infrared transmission, have also been studied (Hollar, 2000). Acoustic communication utilizes a sounder and a microphone. The microphone requires little power, while the sounder power is comparable to low power RF devices; these features are very attractive from a power saving perspective. Some disadvantages in acoustic communication include the following: the surrounding background noise limits the communication quality, and the atmosphere attenuates the signal power over large distances. Moreover, its slow wave propagation speed becomes a serious problem for timing-critical applications. Laser communication can be classified as either passive or active communication, according to whether the laser signal is generated through a secondary source or not. Passive devices receive signals through a laser, and send back data depending on the received command by modulating mirrors located on the passive device. Without the necessity of generating a laser signal, the passive device consumes a small amount of power. Active laser communication points an on-board laser transmitter toward a receiver and transmits data. Long communication ranges, up to 21.4 km (Hollar, 2000), have been reported. Because of the short wavelength, i.e., 100 to 350 nm for lasers as compared to 0.01 to 30 m for RF, and the resultant small divergence, information can be transmitted at lower power than RF. However, searching for the correct receiver direction can take a significant amount of time, and the point-to-point communication is fixed unless the transmitter is redirected through a time- and power-consuming receiver search. Infrared communication uses infrared light with wave lengths ranging from 700 nm to 1 mm. IrDA, an infrared wireless communication technology, consumes significantly less power than RF transceivers (Miller, 2001). The IrDA, however, uses a relatively narrow focused beam which usually requires that the transmitter and the receiver be carefully aligned with each other.

Most of the smart sensors to date employ RF-based wireless communication. As opposed to the laser and infrared communication, RF communication is omni-directional, and does not need direct line-of-sight. In the United States, the Federal Communications Commission (FCC) has allocated the three RF bands around 900 MHz, 2.4 GHz, and 5.0 GHz (0.33 m, 0.125 m, and 0.06 m, respectively, in terms of wavelength) as unlicensed industrial, scientific, and medical (ISM) bands. Many of today's wireless technologies operate on these ISM bands. Small RF chips, which require low power and have few external components, are available from many suppliers. For example, the CC1000 from Chipcon, Inc. (2007) consumes only 8.6-25.4 mA for transmission and 9.6 mA for reception at 868 MHz with a 3 V supply voltage. However, RF chip communication speed is typically limited; the throughput of the CC1000 is only 38.4 kbps. Omni-directional communication with moderate power consumption suits a dense array of smart sensors. RF communication in the ISM band has often been adapted for applications needing small amounts of data transfer such as temperature monitoring.

The RF link also has a limited communication range. As the distance between transmitter and receiver gets longer, more data is lost. Communication between two nodes can be achieved by either home runs, where two nodes communicate with each other directly, or hopping, where nodes between two communication endpoints work as relay nodes.

For communication in a network of a large number of smart sensors, hopping communication has advantages in terms of power. Power requirements for RF communication are proportional to the distance raised to some power. Depending on interference effects, the power requirement could be proportional to the fifth power of distance (Zhao & Guibas, 2004). Also, maximum transmission power for the ISM bands is restricted to be less than 1 W by FCC, limiting the communication range. For these reasons, smart sensors spatially distributed over large areas, as is the case for SHM of bridges and buildings, need to communicate with each other through hopping. However, this simple analysis on power consumption does not include all the communication related tasks affecting the total power consumption in a system. For example, waiting for possible incoming messages in a listening mode also consumes power. Additionally, latency and robustness concerns accompany a large number of relay nodes. Optimal design for communication range is sought with these considerations.

Such RF communication links result in the following characteristics of smart sensors: (a) Wireless communication is not as fast as wired communication; (b) Data loss is inherent to exchange of communication packets; and (c) The network route is not physically fixed, which enables self-configuration and self-healing of networks, and dynamic routing with the help of the on-board microprocessor.

Several standards have provided RF communication specifications. For example, the Bluetooth standard provides the radio link, baseband link, and the link manager protocol (the Open Systems Interconnection reference model; Zimmermann, 1980) for low-power Wireless Personal-Area Networking (WPAN). The expected application is short-range, low-power voice and data communication, such as Personal Digital Assistant (PDA), mobile phones, and laptops. Bluetooth is based on proximity networking at 2.4 GHz, with an expected communication range of around 100 m at an output power of 100 mW, although 1 and 2.5 mW specifications are also available. Data throughput is 723.1 Kbps. IEEE 802.11 is a Wireless Local Area Networking (WLAN) specification. The specification includes robust, Ethernet-style data networking components, such as a Transmission Control Protocol/Internet Protocol (TCP/IP) stack. At the expense of power consumption, this technology possesses communication ranges larger than 100 m and a throughput of up to 54 Mbps. For example, an 802.11b compatible transceiver chip, MAX2820 from Maxim Integrated Products, Inc. (2007), consumes more than 200 mW, while other external components also draw current. IEEE 802.11 technology is well-suited for devices with stable power supplies and high data rate requirements. The IEEE 802.15.4 standard defines the physical and Media Access Control (MAC) layer protocols for WPAN with low data rates but needing a very long battery life. The data rate is 250 Kbps. ZigBee is an industry consortium to promote this standard. The CC2420 from Chipcon, Inc. implements the IEEE 802.15.4 communication standard, consuming only 8.5-17.4 mA for transmission and 19.7 mA for reception.

These RF communication standards bring significant benefit to smart sensor applications, though some applications may need customization. By using the standards, users can avoid the necessity to design all the layers of the OSI reference model (Zimmermann, 1980) from scratch. Users can choose a standard which best fits each application. Also the standards allow independently developed systems to communicate with each other easily. The available standards, however, do not necessarily offer the best RF solution to a specific smart sensor application. Users may want to customize the standards if available standards do not offer a suitable solution at the expense of losing compatibility with other systems.

4. Battery powered

Smart sensors are frequently powered by local batteries, particularly when other power sources are not easily reached. Even when power is available, such as in a building, tethering a large number of smart sensors to power sources is expensive and reduces the merits of wireless communication. There have been several attempts to harvest energy at sensor nodes locally, for example, from mechanical vibration (Rahimi et al., 2003). Vibration energy for civil infrastructure is, however, typically contained in the low frequency range, where electric energy is difficult to harvest. Currently available smart sensors rely on the local battery power supply, which has finite capacity and finite life. Because replacing or recharging batteries in smart sensors installed on a structure may not be a trivial task, power saving is a major concern. This finite energy source problem imposes a strict constraint.

5. Low cost

Smart sensors primarily composed of MEMS and other integrated circuits (IC) have the potential to be produced at a low cost as well as being small in size. This feature of smart sensors, combined with inexpensive installation cost due to wireless communication, enables numerous smart sensors to be densely distributed over civil infrastructure, thus offering the potential to capture the structure's state in detail, and drawing us closer to realizing the dream of ubiquitous sensing.

2.1.2 Smart sensors to date

Some of the first efforts in developing smart sensors for application to civil engineering structures were presented by Straser and Kiremidjian (1996, 1998) and Kiremidjian et al. (1997). In their work, a wireless sensor unit was constructed consisting of a microprocessor, radio modem, data storage, and batteries. To reduce the battery consumption, the smart sensor could be either in a waiting mode or an operational mode. Since these first efforts, numerous researchers have developed smart sensing platforms. Lynch and Loh (2006) cited over 150 papers on wireless sensor networks for SHM conducted at over 50 research institutes worldwide. These platforms can be grouped into two primary categories: proprietary and nonproprietary

Many proprietary platforms have been developed by individual research groups, including the work of Straser and Kiremidjian. Other examples include Mason et al. (1995), Bult et al. (1996), Agre et al. (1999), Aoki et al. (2003), Basheer et al. (2003), Kawahara et al. (2003), Kottapalli et al. (2003), Shinozuka (2003), Wang et al. (2003), Casciati et al. (2004), Sazonov et al. (2004), Farrar et al. (2005), and Lynch (2006), all of whom designed their smart sensor hardware using commercial-off-the-shelf (COTS) components. Each of these papers cited shortcomings in previously developed smart sensors as motivation for the development of their own hardware. While the individual researchers advanced the state-of-the-art in smart sensor technology, progress was slow due to the lack of coordination and leveraging of efforts.

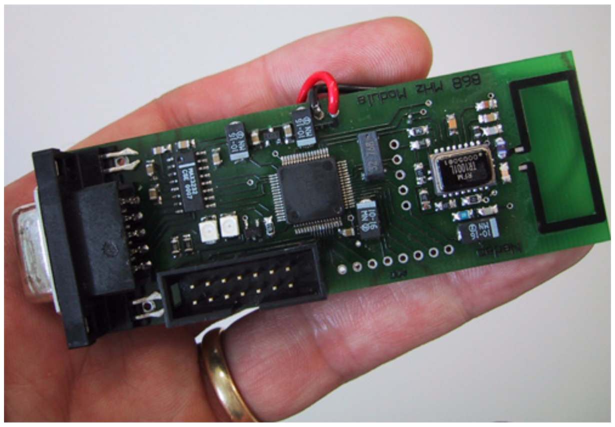


Figure 2.1. EYES project sensor prototype.

Coordinated research efforts, such as was done in the EYES project on self-organizing and collaborative energy efficient sensor networks (EYES, 2006; Law et al., 2002) out of Europe, leveraged individual contributions. The EYES project aimed to develop architecture and technology that will enable the creation of a new generation of self-organizing and collaborative sensors. These sensors will be capable of effectively networking together, to provide a flexible platform to support a large variety of mobile sensor network applications. EYES will make use of the effort invested in the DataGrid project (DataGrid Project, 2006), which is to build the next generation of computing infrastructure, providing intensive computation and analysis of shared large-scale databases. This EYES project includes more than 12 WorkPackages (WP) that deal with middleware, applications, and management. The architecture of EYES has two levels. The first level deals with the sensors and the network, i.e., internal sensor architecture, distributed wireless access, routing protocols, reliable end-to-end transport, synchronization and localization of nodes. The second level provides distributed services to the application, deals with information collection, lookup, discovery and security. Figure 2.1 shows a sensor prototype from the EYES project. Though this coordinated effort provides a more effective framework than the individual approaches, the work is still proprietary in nature, limiting the technical information available to the public.

Commercialized smart sensors also offer the possibility of leveraging contributions among a large number of users. The availability of wireless sensor hardware, especially to application researchers who do not specialize in hardware design, has prompted smart sensor applications in various areas (e.g., machinery monitoring, building HVAC control, etc.). Companies, such as Dust Networks (Dust Networks, 2007), Microstrain (Microstrain, Inc. 2007), Millennial Net (Millennial Net, 2007), Sensametrics (Sensametrics, 2007), and Sensicast Systems (Sensicast Systems, 2007) have designed their own hardware, middleware, and application software, and provide associated support. These sensors tend to emulate wired sensors where raw data from sensor nodes is transmitted to the base station, which is not scalable to a large number of smart sensors due to the limited bandwidth. The lack of scalability becomes substantial, especially when data is acquired at a high sampling ratio, as is the case for civil engineering applications.

The specialized and proprietary nature of the commercial smart sensors has inhibited free hardware/software development for civil engineering applications.

The first available open hardware/software research platform, which allows users to customize hardware/software for a particular application, was the Berkeley Mote. Under substantial support of the U.S. Defense Advanced Research Projects Agency (DARPA), researchers at the University of California at Berkeley have developed the open platform. The main objective is to create massively distributed sensor networks, which consist of hundreds or thousands of sensor nodes; these nodes have been termed as Smart Dust or Mote. The goal is to have fully autonomous sensor nodes that are a cubic millimeter in size.

A family of Berkeley Mote hardware has been developed since the first Mote, termed COTS Dust (Hollar, 2000), arrived, incorporating communications, processing, sensors, and batteries into a package about a cubic inch in size. The generation of the Mote following COTS Dust was the Rene, developed in the summer of 2000. The third generation of Mote, the Mica, was released in 2001 (Hill & Culler, 2002), having improved memory capacity and using a faster microprocessor (Atmega128L, 4 MHz). Subsequent improvements to the Mica platform resulted in the Mica2, Mica2Dot, and MicaZ. Sensors for the Micas are designed on stackable sensor boards separated from the main radio/processor board, so that users can switch between several sensor boards according to their application. The next generation of the Berkeley Mote is the Telos platform, which seeks to achieve three major goals: ultra-low power operation compared with previous Motes, ease of use, and robust hardware and software implementation (Polastre et al., 2005). The Berkeley Mote hardware schematic is publicly available; researchers can follow and customize the design, and expand its sensing modality with their own sensor boards. Moreover, these sensors have been made commercially available through Crossbow Technology, Inc. (2007). The hardware specification of representative Berkeley Motes is summarized in Table 2.1. Note that the limited radio bandwidth, small RAM size, and low resolution Analog-to-Digital Converter (ADC) of these Motes, as well as the modest microprocessor power can be a restraining factor for intensive sensing/ calculation applications like SHM.

Table 2.1. *Smart Sensor Specifications*
(The power consumption is estimated based on associated components' data sheet.)

	MicaZ	Mica2	Mica2dot	Telos	Imote2
Microprocessor	ATmega128L	ATmega128L	ATmega128L	TIMSP430	XScalePXA271
Bus size (bits)	8	8	8	16	32
Clock speed (MHz)	7.373	7.373	4	6.717	13-416
Program flash (bytes)	128 k	128 k	128 k	48 k	32 M
EEPROM (bytes)	4 k	4 k	4 k		
RAM (bytes)	4 k	4 k	4 k	10 k	256 k+ 32 M external
Active power (mW)	24 @ 3 V	24 @ 3 V	15 @ 3 V	10 @3 V	44 @ 13 MHz 570 @ 416 MHz
Sleep power (μW)	75	75	75	8	100
ADC resolution (bits)	10	10	10	12	N/A
ADC channels	8	8	6	8	N/A
Nonvolatile storage (bytes)	512 k	512 k	512 k	1,024 k	32 M (Identical to the program flash)
Memory write power (mW)	45	45	45	60	120
Memory read power (mW)	12	12	12	12	108

Table 2.1. *Smart Sensor Specifications (Continued)*
(The power consumption is estimated based on associated components' data sheet.)

	MicaZ	Mica2	Mica2dot	Telos	Imote2
Radio Chip	CC2420	CC1000	CC1000	CC2420	CC2420
Radio frequency (MHz)	2400-2483.5	315/433/915	315/433/915	2400-2483.5	2400-2483.5
Data rate (kbps)	250	38.4	38.4	250	250
Outdoor range (m)	100	300/300/150	300/300/150	100	100
Transmission power at 1 mW power (mW)	52	31	31	52	52
Reception power (mW)	59	22	22	59	59
Typical battery option	2 x AA	2 x AA	CR2354	2 x AA	3 x AAA
Typical capacity (mAh)	2 x 1500-2000	2 x 1500-2000	560	2 x 1500-2000	3 x 700
Typical supply voltage	3	3	3	3	4.5
Size (mm)	58 x 32 x 7	58 x 32 x 7	25 x 6	65 x 31 x 6	48 x 36 x 7
Available Sensor	Light Temperature Humidity Barometric pressure Accelerometer GPS Microphone Sounder Magnetometer	Light Temperature Humidity Barometric pressure Accelerometer GPS Microphone Sounder Magnetometer	Light Temperature Accelerometer	Light Temperature Humidity	Light Temperature Humidity Accelerometer

While a number of sensor boards for the Berkeley Mote have been designed for generic applications and are commercially available (see Table 2.1), available sensors are not optimized for use in civil infrastructure applications. Acceleration and strain are among the most important physical quantities to judge the health of a structure. While acceleration measurements are essential to obtain global responses of a structure, structural strains provide an important indicator of local structural behavior. Studer and Peters (2004) demonstrated that multiscale sensing yields better results than single-scale measurements for damage identification. Available sensor boards have accelerometers, though their applicability to civil infrastructure is limited. Ruiz-Sandoval (2004) examined an acceleration sensor board, the MTS310CS, from the civil engineering perspective; he found the accelerometer's performance is deficient, particularly in the low frequency range, and developed a new sensor board, 'Tadeo,' with high sensitivity and low noise level. The sensor boards of the Berkeley Mote can be conveniently customized according to the applications.

The Berkeley Mote uses a open-source operating system, TinyOS, which is available online (http://www.tinyos.net). TinyOS is a component-based operating system designed for sensor network applications on resource-constrained hardware platforms, such as the Berkeley Mote. More specifically, it is designed to support concurrency intensive operations such as are required by networked sensors with minimal hardware requirements; TinyOS fits in about 4 kB of memory space. TinyOS is, however, not a real-time operating system. TinyOS has only two levels of priority, each of which corresponds to hardware interrupt and tasks. Users cannot assign priority arbitrarily to specific tasks or assume highly precise scheduling. This limitation needs to be well considered in designing a system using TinyOS.

A wide community uses TinyOS to develop and test various algorithms and protocols. The operating system has been ported to over a dozen platforms employing numerous sensor boards. The autonomous characteristics of the smart sensor can be realized by developing programs under TinyOS and then running these programs, along with the operating system, on the on-board microprocessor. Similar software for the Berkeley Mote and other platforms includes MANTIS (Bhatti et al., 2005) and EmStar (Girod et al., 2004). Because of the open nature, numerous researchers have contributed to TinyOS enhancement; users can take advantage of the wealth of previous studies.

Open-interface smart sensor platforms developed by companies, such as Ember (2007) and Intel (2007), have similar potential. Users have access to the programming interface and can program the sensor nodes as needed. Ember nodes must be programmed on their own OS, but users can implement necessary functionalities through an open interface. Ember proposed its own algorithm to construct the mesh-network. Though the ready-to-use mesh networking is advantageous for some applications, the patented nature of networking may result in an inflexible network system design. Intel recently developed the Intel Imote2, which runs TinyOS or Linux (Adler et al., 2005). The Imote2 provides enhanced computation and communication capabilities that allow resource-demanding sensor network applications, such as SHM of civil infrastructure, to be supported, while low-power operation and small physical size are still among the objectives. The wealth of studies using TinyOS can be implemented directly to the Imote2; this commercial open-interface platform will further advance the state-of-the-art in smart sensor technology, especially for SHM application.

2.2 Middleware services

Prior to implementing SHM algorithms on a smart sensor network, many issues intrinsic to the wireless sensor must be addressed. These issues include time synchronization, routing, and reliable communication. These issues are not unique to civil engineering applications, but are common for many smart sensor applications, including environmental and habitat monitoring and sniper detection (Simon et al., 2004). However, services to address these issues are not in general provided by operating systems. Middleware services usually address these issues. Wireless sensor network middleware services tend to be more application specific, as compared to those for PCs and servers, because the limited resources of smart sensor networks (e.g., power, memory, and bandwidth) need to be optimally utilized according to the specific applications (Romer et al., 2002). Requirements for each middleware service need to be provided clearly; users choose or develop middleware services which suit their requirements. Middleware services relevant to this research are reviewed herein.

Because smart sensor nodes have their own clock, these nodes do not share global time. This lack of a global clock is problematic for SHM applications. For example, if modal analysis is conducted without accurate time synchronization, the identified mode shape phase will be inaccurate, possibly falsely indicating structural damage. To address this issue, several time synchronization techniques have been proposed so far. Reference Broadcast Synchronization (RBS; Elson et al., 2002), Flooding Time Synchronization Protocol (FTSP; Maroti et al., 2004) and Timing-sync Protocol for Sensor Networks (TPSN; Ganeriwal et al., 2003) are among the well-known synchronization methods. Lynch et al. (2005) implemented the RBS and reported a maximum time delay of 0.1 seconds. Mechitov et al. (2004) implemented FTSP as a part of wireless data acquisition system for SHM. Their system can maintain better than 1ms synchronization accuracy for a long period of time. When clock rate differs from node to node, clock drift is not negligible. Synchronization needs to be applied periodically, or the difference in clock rates needs to be estimated and compensated. Also packet transfer associated with time synchronization is subject to packet loss. Reliability of synchronization in a lossy communication environment is also an issue. Though accuracy and reliability of time synchronization depends on the method and the hardware/software architecture, synchronization better than 1 ms has been claimed by many researchers (Elson et al., 2002; Maroti et al., 2004; Mechitov et al., 2004) and is becoming a reasonable assumption.

Smart sensors can build several network topologies, including (a) the star topology where one coordinator node directly communicates to other nodes, (b) the mesh topology where multiple routing nodes communicate with each other to make a redundant network, and (c) the cluster tree topology where routing nodes are organized in a tree to connect all the nodes, usually without redundant connections. This classification is consistent with the ZigBee standard. The star topology is not geometrically scalable because all of the nodes need to be within the coordinator's communication range. Ember (2005) employed the mesh network to monitor water treatment plants, using their ZigBee chipset. The first generation Intel Mote (Kling, 2003; Kling et al., 2005) employed the cluster tree topology utilizing Bluetooth technology. Mechitov et al. (2004) employed the tree topology with Mica2s. By refreshing the topology periodically, the sensor networks are resistant to communication link failures. Most of the current civil engineering applications using smart sensors utilize the star topology. For mesh or cluster tree topologies, the routing path needs to be optimized. To collect data efficiently and reliably at the base station, Mechitov et al. (2004) set the path length as the primary criterion for establishing the tree structure, with link quality being the secondary criterion.

Packet loss intrinsic to wireless communication needs to be addressed. Some data need to be delivered reliably while others need to be delivered with better than a certain probability. Although packet loss is inevitable, the loss of data can be avoided or reduced. Sending a packet multiple times increases the possibility of successful delivery. Furthermore, a packet can be repeatedly sent until acknowledgment is received. This approach guarantees successful communication unless two nodes are completely out of the communication range of each other or one of the nodes fails. Reliable communication with acknowledgment messages was implemented on the Mica2 platform (Mechitov et al., 2004). Also, standards such as Bluetooth and IEEE 802.11, provide their own specifications to deal with communication errors. Successful delivery can be guaranteed or increased at the expense of extra packets to exchange, larger header size, and/or extra computation.

Because RAM is one of the power-consuming components on a smart sensor, the size of RAM is usually small, sometimes necessitating virtual memory or other solutions. Applications such as SHM deal with large response data records and need a large amount of memory. For example, a modal analysis method, Eigensystem Realization Algorithm (ERA; Juang and Pappa, 1985) employs singular value decomposition of two Hankel matrices, whose size could be much larger than the RAM size; a 50 x 50 rather small Hankel matrix consisting of 8-byte, double precision matrix elements needs 20 kB of RAM, while hundreds by hundreds Hankel matrices are not uncommon. Mica2's 4 kB of RAM memory is apparently insufficient for this SHM application. Kwon et al. (2005b) proposed ActorNet which employs virtual memory on the Mica2. In addition to the virtual memory, large physical memory can be a solution. For example, the Imote2 possesses 32 MB of RAM at the expense of larger power consumption.

Because of efficient power usage considerations, TinyOS employs nonblocking I/O, which makes programming code complicated. The nonblocking I/O system does not wait for a return value after it calls a function. A called task is posted for execution, and the main thread keeps running. Once the task is executed, the completion is signaled. Kwon et al. (2005b) proposed converting nonblocking I/O to blocking I/O to make the program development more efficient. This nonblocking I/O is not a critical problem to be solved, but this middleware service will increase the productivity of the inexperienced developers.

Though many middleware services have been proposed, studied, and implemented, their performance has not been carefully examined from the SHM perspective. Moreover, the middleware requirements for SHM are unclear. Based on SHM application requirements, appropriate middleware services will need to be adopted, modified, or developed.

2.3 Structural Health Monitoring

Structural Health Monitoring (SHM) strategies measure structural response and aim to effectively detect, locate, and assess damage produced by severe loading events and by progressive environmental deterioration. Structural response reflects the structural condition as well as the excitation force. By analyzing the response data, SHM strategies are expected to reveal structural condition, such as the damage existence. SHM has seen intense research efforts in mechanical, aerospace, and maritime, as well as civil engineering applications. In particular, condition-based machinery maintenance utilizing vibration monitoring has seen many applications (Al-Najjar, 2000; Collacott, 1977). Once damage is detected, detailed Non-Destructive Testing (NDT), repair, and/or suspension of service follow. Well-known NDT techniques include visual inspection, eddy current testing, acoustic emission, ultrasonic testing, and radiographic inspection (Gros, 1997). SHM leads to well-organized maintenance of structures.

Efficient and effective maintenance is among motivations to introduce SHM strategies for various applications. The Electric Power Research Institute (EPRI) has successfully demonstrated the use of diagnostic technology at its Eddystone Station fossil-fuel power plant outside of Philadelphia. Since the 1987 installation of a dual-purpose vibration monitor and rotor-crack detector for bearing wear at the plant, EPRI says it has saved $250,000 in costs for teardown of machinery. In military applications, SHM is expected to reduce maintenance and manpower requirements by approximately 20 to 40 percent, increase combat sorties by 25 percent, and reduce the complexity of the logistics trail by 50 percent, compared to current military strike aircraft (Becker et al., 1998). SHM in these fields has been shown to be a practical and an attractive solution for maintenance.

From a civil engineering perspective, SHM is expected to provide an efficient and effective tool for management of infrastructure. Civil infrastructure is a valuable asset, which keeps the economy and people's life running. For example, a long-span bridge such as the Akashi-Kaikyo Bridge in Japan costs billions of dollars, while the Golden Gate Bridge in California has about 40 million crossings per year (Golden Gate Bridge, Highway and Transportation District, 2005). These structures, however, deteriorate with age. Many of the 1,100 major long-span bridges in the U.S. are over 50 years old and several notable ones are over 100 years old. More than 800 of the long-span bridges in the National Bridge Inventory are classified as fracture-critical (Pines & Aktan, 2002). The structural integrity and serviceability of these structures are not necessarily apparent. Because failure of such civil infrastructure can have significant negative impacts on society at large, structures need to be managed based on the understanding of their current structural conditions. The current practice, visual inspection, is expensive both in time and cost. For example, the biennial visual inspection of a major bridge such as the Brooklyn Bridge in New York is reported to last for over 3 months at a cost of $1 million (Dubin & Yanev, 2001; Pines & Aktan, 2002); note that the reliability of such visual inspection is being examined (Moore et al., 2001). SHM potentially offers better and more efficient understanding of structural conditions.

In addition to structural deterioration due to aging, SHM can be employed to estimate structural damage due to severe loading events, such as earthquakes, hurricanes, or tornados, facilitating more timely recovery from such disasters. It is imperative that emergency facilities and evacuation routes, including bridges and highways, be assessed for safety. Traditional detailed assessments can be significantly expensive and time-consuming, as was seen after the 1994 Northridge earthquake with the numerous buildings that needed to have their moment-resisting connections inspected. Additionally, structures that are internally, but not obviously, damaged in an earthquake may be in great danger of collapse during aftershocks; structural integrity assessment can help to identify such structures to enable evacuation of building occupants and contents prior to aftershocks. SHM enhances safety and reliability of civil infrastructure after such disastrous events.

Though the necessity is clear, the features of civil infrastructure pose difficulties to SHM. These features include large scale, a myriad of elements, one-of-a-kind designs, low natural frequencies, structural redundancy, nontrivial test excitation, challenging input force estimation, long and continuous service time, and variable environmental conditions. For example, changes in resonant frequencies of redundant structures, which often imply structural damage, can be insignificant as compared to frequency shifts due to changes in ambient conditions, such as temperature and support stiffness (Aktan et al., 1994; Chowdhury, 1990; Farrar et al., 1994; Salawu, 1997; Tang & Leu, 1989). Many researchers have been working to overcome these difficulties.

Doebling et al. (1996) reviewed research on vibration-based damage identification and health monitoring. The cited papers were categorized based on the method, such as frequency change, mode shape change, mode shape curvature, flexibility, matrix update, nonlinear methods, and neural network-based methods. Sohn et al. (2003) reviewed papers on SHM published between 1996 and 2001. A summary of statistical approaches for damage detection was also included. Other review papers include Salawu and Williams (1995), Salawu (1997), Doebling et al. (1998), Doebling and Farrar (1999), and Farrar et al. (2003). Major algorithms for SHM are briefly described below.

Most of the SHM methods mentioned above employ modal analysis to obtain modal parameters such as natural frequencies, damping ratios, and mode shapes. A number of modal analysis methods have been proposed. Peak-picking is a simple frequency domain modal analysis method. Improvements by incorporating the coherence function and using frequency domain decomposition (Brincker et al., 2001) have also been reported. Maximum likelihood identification, which includes curve-fitting, estimates modal parameters by minimizing an error norm in the frequency domain. Time domain modal analysis methods include the complex-exponential method (Maia & Silva, 1997); Ibrahim time domain method (Ibrahim & Mikulcik, 1977); ERA (Juang & Pappa, 1985); and stochastic subspace identification (Hermans & Auweraer, 1999). Identified modal parameters are further analyzed for damage detection.

Natural frequency changes reflect structural conditions, offering clues to estimate the structural conditions. By analyzing the natural frequency change, the cause of structural damage is expected to be revealed. For example, the tension in the stay cables of cable-stayed bridges can be estimated by natural frequency measurement (Gardner-Morse & Huston, 1993). The frequency changes, however, may have low sensitivity to damage as mentioned earlier. Begg et al. (1976) pointed out that cracks have little influence on global modes and suggested that local high-frequency bending modes of the individual members provide a better indication of cracking. Srinivasan and Kot (1992) conducted experiments on a shell structure and found that the resonant frequencies of the shell structure were insensitive to damage. Also, frequency shifts depend on each mode. Salawu (1997) mentioned that the stress induced by modal deformation is minimal at modal nodes, where modal displacement is zero; a small change in a particular modal frequency could mean that the defect may be close to the modal node. Hearn and Testa (1991) developed a damage detection method that examines percentage changes in natural frequencies. To relate observed frequency changes or percentage changes to structural damage, these methods require theoretical structural models as well as a damage model or sensitivity analysis; these model estimations and the analyses are not straightforward (Salawu, 1997).

Fluctuation in damping values is much larger than that in frequencies (Williams & Salawu, 1997) and can potentially be a damage indicator. High damping would suggest more energy dissipation mechanisms, indicating the possibility of cracks in the structure (Morgan & Osterle, 1985). Changes in damping values up to 80 percent were reported by Agardh (1991). Williams and Salawu (1997), however, pointed out that damping properties are the most difficult to model analytically and can only be realistically obtained through vibration tests. Nashif et al. (1985) and Sun and Lu (1995) give comprehensive discussions. Relatively large estimation and modeling errors are against the usage of damping for SHM.

Mode shape information has also been investigated. West (1984) compared mode shapes of a space structure, using Modal Assurance Criterion (MAC), before and after exposure to loading. The mode shapes were partitioned and changes in the local MAC values along the mode shapes were used to locate the damage. Fox (1992) stated that mode shape changes are insensitive to damage and “Node line MAC,” a MAC variant based on measurement points close to a node point for a particular mode, is a more sensitive indicator of damage. Pandey et al. (1991) demonstrates that absolute changes in mode shape curvature, calculated through difference analysis of displacement mode shapes, can be a good indicator of damage. Chance et al. (1994) and Nwosu et al. (1995) used measured strains instead of the calculated curvature, reducing the numerical error associated with difference analysis. Salawu and Williams (1994), however, pointed out that the selection of which modes are used in the analysis is an important factor, and curvature changes do not typically give a good indication of damage using experimental data. Mode shape phase has also been reported to be a possible damage indicator. Mode shapes of proportionally damped dynamic systems are theoretically purely real, i.e., they are aligned within a plane, resulting in either 0 or π radian phase. Structural damage, especially damage with friction type mechanisms such as loose bolt connections, often introduces nonproportional damping, which results in complex mode shapes, i.e., the phases will differ from 0 and π. These phase changes can potentially indicate structural damage, although structural damage is not the only cause to change the phase of mode shapes (Nagayama et al., 2005). While some mode shape related indicators reflect structural damage, the analysis to quantitatively relate the mode shape information to structural damage is yet to be completed.

The dynamically measured flexibility matrix, calculated from the mass-normalized mode shapes and modal frequencies, has been employed to determine structural damage. By interpreting the physical meaning of the flexibility matrix, Toksoy and Aktan (1994) observed that anomalies in the flexibility matrix can indicate damage even without a baseline data set. Gao et al. (2005) reported that the flexibility matrix is estimated with moderate accuracy using only a few lower modes, as opposed to the stiffness matrix (the inverse of the flexibility matrix), which needs almost all of the modes. Though some researchers insist that higher modes better indicate the damage due to their insensitivity to support conditions and high sensitivity to local damage (Alampalli et al., 1992; Biswas et al., 1990; Chowdhury, 1990; Lieven & Waters, 1994), estimation accuracy of these modes is arguable. If adequate for the purpose, lower modes with more accurate estimation would better suit SHM. The ability of the flexibility matrix to be estimated with only a few lower modes is, therefore, advantageous. He and Ewins (1986), Lin (1990), and Zhang and Aktan (1995) further studied flexibility-based SHM strategies. Deficient points of these techniques include that the error due to the unmeasured modes and flexibility change due to damage cannot be clearly separated, and that structural constraints such as symmetry and connectivity have not been fully incorporated, though Doebling (1995) conducted research to tackle this problem.

Matrix update methods offer another class of SHM methods. Structural characteristics such as mass, stiffness, and damping are represented in matrix form and updated to be consistent with the observation of the structure under consideration. As Doebling et al. (1996) described, the objective function is numerically optimized to update the matrices under constraints such as matrix sparsity, connectivity, symmetry, and matrix positivity. Doebling et al. (1996) categorized matrix update methods as closed-form optimal matrix update, sensitivity-based update, eigenstructure assignment, and hybrid matrix update. Zimmerman and Kaouk (1994) introduced minimum rank perturbation theory (MRPT) to the optimal matrix update problem; this approach has been published extensively (Kaouk & Zimmerman, 1994, 1995; Zimmerman & Simmermacher, 1995; Zimmerman et al., 1995). The limitation of these methods is that the rank of the perturbation is always equal to the number of modes used in the computation of the modal force error. Also, the number of degrees-of-freedom (DOFs) and type of FEM models can affect the final results.

Wavelet transforms and analytical signals have been among popular data analysis tools. Wavelet transforms are often used to extract features from complicated data. Staszewski (1998) summarized the application of wavelet analysis for SHM. While wavelet transforms provide flexible and powerful data analysis tools that can be used in combination with other methods, the transforms do not possess a clear physical meaning, as opposed to the Fourier transform. Feldman and Braun (1995) used an analytical signal, which is a combination of the actual signal and its Hilbert transform, to get an instantaneous estimate of the modal parameters. The analytical signal may not work when more than one modal component exists in the signal. Features of nonstationary processes are often analyzed with these tools.

Nonlinearity of structural behavior due to structural damage has also been studied. Crack opening and closing was investigated by Actis and Dimarogonas (1989) and Krawczuk and Ostachowicz (1992). Lin and Ewins (1990) contemplated response level dependencies of structural properties utilizing modal properties measured at different response level. Sohn et al. (2007) proposed damage diagnostics based on the concept of time reversal acoustics and consecutive outlier analysis to detect nonlinearity between a pair of sensors.

Sohn et al. (2003) emphasized the importance of statistical models to enhance the SHM process. Statistical model development can be classified as supervised learning and unsupervised learning. Sohn et al. (2003) discussed supervised learning methods such as neural networks (Bishop, 1994; Nakamura et al., 1998; Zhao et al., 1998), genetic algorithms, support vector machines (Vapnik, 1998), outlier detection, and hypothesis testing. These supervised learning algorithms are first trained by structures of known properties or FEM models, and then applied to the structures to be tested. Therefore, this algorithm may require significant training with the undamaged structure as well as structures damaged in one or more of the failure modes. While this training may be possible for structures produced in larger lots (e.g., aircraft), the training is nontrivial for unique structures (e.g., buildings and bridges). On the other hand, unsupervised learning may detect abnormalities, or damage presence, but locating damage and assessing the severity need further theoretical development. Sohn et al. (2002) proposed to combine AR-ARX model, nonlinear principal component analysis, and statistical analysis to distinguish environmental and structural changes. Computer simulation and experimental results proved that this method can detect damage. By placing sensors at each DOF and applying this procedure for each sensor, Sohn et al. (2002) successfully located damage of an 8 DOF experimental model under environmental change.

Bernal (2002) proposed a flexibility-matrix-based damage localization method, the Damage Locating Vector (DLV) method. The DLV method has advantages in that structural responses do not need to be measured at all the DOFs, though small numbers of sensors result in limited damage detection capability. A set of load vectors, designated as DLVs, were computed from the change in the flexibility matrix. The flexibility matrix can be dynamically estimated. When the DLVs are applied as static forces on the undamaged structure, the stress field in the structure bypasses the damage areas. This unique characteristic of the DLVs can be employed to localize damage in the structure.

Although many methods have been proposed, none have proven to be sufficient for full-scale application. Many of the SHM algorithms mentioned above have been shown to detect damage well when a large enough number of modes are accurately measured at all the DOFs. Even the DLV method, which does not require response measurements at all the DOFs, performs poorly when used with a small number of sensors. The more sensors used, the more information about structures SHM is expected to give. However, structures are usually large and have numerous DOFs; accurate and thorough measurements have been impractical. Sensors have limited accuracy and the associated installation cost, including cabling, has been prohibitively expensive. For example, Lynch and Loh (2006) cited Farrar (2001), which reported that the cost of installing over 350 sensing channels on the Tsing Ma Bridge in Hong Kong was more than $8 million. Celebi (2002) estimated that sensor installation cost including labor, cabling, and recording systems is about $4,000 per sensing channel. The emergence of smart sensors with wireless communication capabilities, as described in the following section, offers the possibility of SHM with dense measurements.

2.4 Attempts toward SHM using smart sensors

2.4.1 Research attempts

Several SHM applications with smart sensors have been reported using scale models. Demonstrated tasks include data acquisition with a single wireless node, synchronized data acquisition with multiple nodes, on-board data processing, etc. Tanner et al. (2002, 2003) embedded data processing on a smart sensor unit. A Mica node was programmed to measure acceleration responses of a beam on both sides of its bolted joint and then calculate the correlation coefficient of the responses to detect a loose bolt. Lynch et al. (2002) implemented Fast Fourier Transform (FFT) to reveal the five-story building model's response in the frequency domain. Nitta et al. (2005) implemented an AR model on the Mica2 and experimentally verified its validity on a three-story building model. These studies have briefly demonstrated the applicability of smart sensor systems to SHM applications using simplified models.

Full-scale buildings and bridges have also been the subject of smart sensor research. Straser and Kiremidjian (1998) and Lynch et al. (2003) measured structural responses of the Alamosa Canyon Bridge to validate their smart sensors' performance. Galbreath et al. (2003) monitored a highway bridge on the LaPlatte River in Vermont, using Microstrain's wireless strain sensor unit (Microstrain, Inc., 2007). Aoki et al. (2003) measured the acceleration response of a light pole on the Tokyo Rainbow Bridge in Japan. The data was transmitted to a data repository using a WLAN. Chung et al. (2004) installed a DuraNode sensor unit on a pedestrian bridge at the University of California, Irvine. Wirelessly collected data was then analyzed on a PC to give the first three vibration modes. Ou et al. (2005) installed eight Mica nodes in the Di Wang Tower in China. Lynch et al. (2005) installed 15 smart sensor units on the Geumdang Bridge in Korea to measure the forced vibration response. FFT was applied to measurement signals on smart sensor nodes independently, and the Fourier transform results were sent back to the base station. In the geotechnical research field, Chen et al. (2005) proposed to use wireless sensor for MEMS-based vertical seismic array, called Terra-Scope. These research attempts have demonstrated the capability of smart sensors to measure acceleration for full-scale civil infrastructure, though the quality of data was not necessarily examined.

Through the laboratory and full-scale structural applications, benefits in terms of the sensor installation time were reported. Straser and Kiremidjian (1998) reported that installation of the wireless system on the Alamosa Canyon Bridge took 30 minutes, which was five times faster than the cable-based system. Lynch et al. (2003) implemented smart sensor units on the same bridge, and the installation time was half the time to install the cable-based system.

Many researchers have also been working on another approach to SHM with wireless communication technology. The data is acquired using a traditional wired acquisition system and then sent back to a remote data depository using a cell phone, WLAN, or other wireless communication (Karbhari et al., 2003; Mufti, 2003;Oshima et al., 2000). However, this framework does not possess the distributed on-board microprocessors, nor does it eliminate the cabling cost for sensor installation. Other approaches should be taken if densely instrumented measurement is an objective.

2.4.2 Difficulties in using smart sensors for SHM applications

Though many researchers have demonstrated the use of smart sensors for SHM applications, none of them have resulted in a full-fledged SHM system. There are difficulties civil engineers commonly encounter when SHM applications are implemented on smart sensors. For example, many of the demonstrated systems are not scalable to a large number of smart sensors. Some smart sensors cannot acquire reliable measurement data due to deficiencies in the sensor, time synchronization errors, data loss, etc. Major difficulties are summarized herein from an SHM application perspective.

1. Sensor hardware

Though several types of sensors have been implemented on smart sensors (see Table 2.1 on page 12), these sensors do not cover all types of sensors that civil engineers typically need. For example, only a few smart sensors have employed a strain sensor, while strain is one of important physical quantities from which to judge structural condition. Velocity or displacement sensors have not seen smart sensor application. When civil engineers do not find appropriate sensors on smart sensor platforms, sensor boards need to be customized; open source platforms make it easier to customize sensor boards.

While accelerometers are some of the most commonly employed sensors, their applicability to civil engineering applications is not apparent. Users cannot simply assume MEMS accelerometer’s characteristics to be similar to those of conventional accelerometers. Acceleration of shake tables or the response of structure model has been recorded with smart sensors to examine the performance of their accelerometers (Arici & Mosalam, 2003; Casciati et al., 2003; Hou et al., 2005; Kurata et al., 2004; Lynch et al., 2002; Ruiz-Sandoval, 2004; Straser & Kiremidjian, 1998).

Limited sensitivity and high noise floor are sometimes problems with MEMS accelerometers. While acceleration noise density of one of the conventional piezoelectric accelerometers, PCB 393B04 (PCB Piezoelectronics, Inc., 2007), is 0.04 $\mu g/\sqrt{Hz}$, that of MEMS accelerometer used on many smart sensor prototypes, ADXL201 (Analog Devices, Inc., 2007) is 200 $\mu g/\sqrt{Hz}$. The accelerometer on the Imote2 sensor board, LIS3L02DQ (STMicroelectronics, 2007), has a noise density of 50 $\mu g/\sqrt{Hz}$. This noise floor is considered low enough for scale-model dynamic testing using an exciter or shake table; however, applicability to ambient vibration measurement of buildings needs further investigation. Ruiz-Sandoval (2004) and Ruiz-Sandoval et al. (2006) employed an accelerometer with a high sensitivity and low noise level, SD1221 (Silicon Designs, Inc., 2007). This MEMS type accelerometer has the noise density of 5 $\mu g/\sqrt{Hz}$. Ruiz-Sandoval

(2004) indicated that this accelerometer is suitable for SHM applications. Users need to be aware of the sensing characteristics of smart sensors.

Characteristics of accelerometers need to be well-examined, especially in the low frequency range, because major vibration modes of civil infrastructure appear in this frequency range. Natural frequencies of tall buildings, towers, or long bridges can be as low as 0.1 Hz. In terms of vibration amplitude, acceleration in the low frequency range is small, underlining the importance of high resolution and sensitivity of a sensing system. Ruiz-Sandoval (2004) and Ruiz-Sandoval et al. (2006) calibrated their sensor board with special emphasis in the low frequency range. Though many smart sensors with accelerometers have been proposed, only a limited number of acceleration sensor boards can measure low frequency vibration accurately.

In addition to the sensor itself, the ADC, AA filter, and supply voltage regulator also influence the quality of measurement signals. A low resolution ADC degrades measured signals by introducing large quantization errors. The ADC on the Mica2, for example, has only a 10-bit resolution, limiting the dynamic range of sensors. Appropriate lowpass filters are essential to obtain digital signals free of aliasing. The sensor's supply voltage needs to be regulated so that current drawn by the microprocessor, radio device, or flash memory does not destabilize current flow to sensing components. These components need to be carefully designed. Otherwise, the structural information submerged in the measurement signals may not be extracted. Because all of these issues affect signal quality, smart sensor users cannot simply assume the sensing characteristics of a sensor node are the same as those of a sensor component.

Even a dense array of smart sensors is not a rich information source for SHM if physical quantities needed cannot be measured precisely by each smart sensor. Development of sensor boards for SHM applications is still an important research issue as well as calibration of these sensor boards.

2. Data aggregation

Data aggregation for SHM applications often encounters the following three issues: (a) data size is too large, (b) data may be lost during wireless communication, and (c) communication range is limited. Each of these problems is summarized below.

Smart sensors in general are not designed to collect a large amount of data, while SHM applications benefit from data acquired from numerous sensors with high sampling frequencies. Early applications of smart sensors, such as habitat monitoring, handled only a small amount of data on an infrequent basis. On the other hand, SHM applications typically acquires tens of thousands data points, each of which is represented as two- or four-byte data. Sampling frequencies higher than 100 Hz and total sampling time longer than a minute are quite common. In view of the need to handle a large amount of data, SHM applications with smart sensors can be categorized into two groups, neither of which has fully exploited the smart sensor's capability.

In the first group, the smart sensors are employed in the same manner as traditional wired sensors, with all data being collected for processing at a centralized location (see

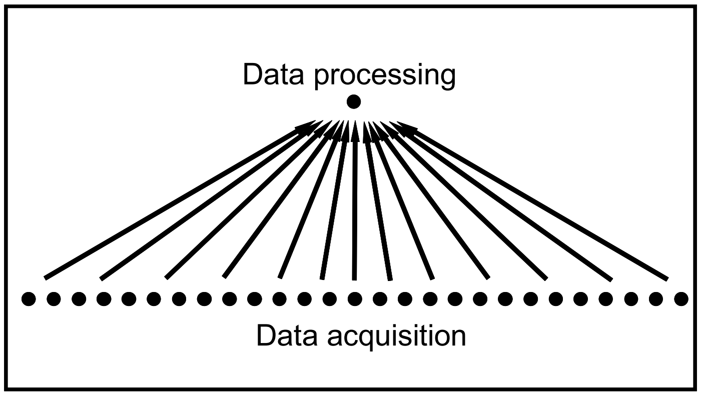


Figure 2.2. Centralized data acquisition approach.

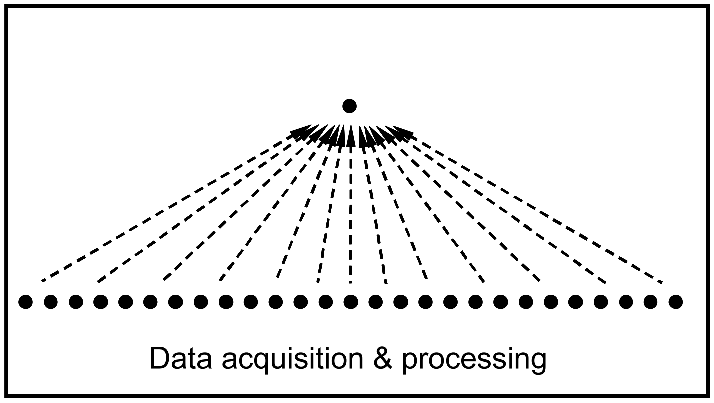


Figure 2.3. Independent data processing approach.

Figure 2.2). Centralized SHM algorithms are then applied to this data. This approach allows for application of a wealth of traditional SHM algorithms reviewed in 2.3 Structural Health Monitoring. As the number of smart sensors increases, however, the measurement data to be centrally collected exceeds the network bandwidth, whether homerun or hopping communication is adopted. The lack of scalability is a serious deficiency of this approach. One approach toward a scalable solution is to have a tiered network. Chintalapudi et al. (2006) utilized lower tier nodes and powerful upper tier nodes. Assuming upper tier nodes have sufficient power, power consumption at lower tier nodes is moderated. The tiered network approach is applicable only when installation of powerful nodes and power supply to these nodes are practical.

The second group, on the other hand, assumes that each smart sensor measures and processes data independently without sharing information among the neighboring nodes as illustrated in Figure 2.3 (Lynch et al., 2005; Nitta et al., 2005; Sohn et al., 2002). Because only the data processing outputs are sent back to the base station, the required communication can be quite small.

Consequently, this approach is scalable to a large number of smart sensors. However, the independent sensor node approach does not utilize available information from neighboring nodes; all spatial information is neglected. For example, information regarding mode shapes cannot be obtained and used in this approach. Information from different type of sensors connected to separate nodes cannot be combined. The inability of this approach to incorporate spatial information limits its effectiveness.

Gao (2005) proposed a Distributed Computing Strategy (DCS) for SHM that offers a scalable approach that can incorporate spatial information. This DCS approach, based on the DLV method (Bernal, 2002), does not need to centrally collect and analyze the measurement data. Instead, the DCS shares data among the neighboring nodes to utilize spatial information. Due to this local data sharing with limited numbers of neighboring nodes, the total amount of data to be transmitted throughout the network is kept modest. Therefore, this SHM strategy is scalable to a large number of sensors densely deployed over large structures. While the DCS does not require measurements at all the DOFs, the method's performance improves with the number of DOFs measured. Computer analysis and experimental validation on a simulated wireless network showed the DCS is a promising SHM scheme. This strategy, however, has not yet been implemented on smart sensors and experimentally verified.

Data loss during wireless communication is also problematic for SHM applications. Wireless communication suffers from packet loss unless lost packets are resent. Kurata et al. (2004) reported data loss during shake table experiments. Many civil engineering applications using smart sensors do not address this data loss problem. Some of them simply ignore lost data while others coincidentally receive all the packets during experiments. However, SHM methods developed so far assume that data acquired at sensors are available for data processing at the base station. The influence of lost data on structural analysis has not been clearly investigated. Mechitov et al. (2004) employed a reliable communication protocol to address this problem. Because acknowledgment packets are sent frequently, the communication speed is slower than communication without acknowledgment. Reliable communication services suitable to transfer large amount of data is expected to advance SHM applications employing smart sensors.

The communication range of smart sensors is usually shorter than the size of civil infrastructure. Centrally collecting data or sending commands to smart sensor nodes on structures involves multihop communication. A routing path usually needs to be determined prior to multihop communication (Mechitov et al., 2004). If a communication path between two arbitrarily selected nodes is required, a very large table to store routing paths needs to be constructed on each sensor node. Application-specific knowledge on communication, such as collecting data to a sink node and dissemination, potentially simplifies routing. Once paths are found, multihop communication can be started. Routing path, packet structure, overhead information, etc. need to be carefully designed.

3. Time synchronization

Dynamic analyses of structures assume that data is synchronized, which is not the default case with smart sensor networks. As described in section 2.2, several synchronization protocols have been proposed. Some of them have synchronization accuracy as good as tens of microseconds. However, the requirement of SHM on time synchronization accuracy is not studied. Based on the fact that natural frequencies of structures used in analyses are usually below 10 Hz, civil engineers may erroneously think that time synchronization error smaller than a millisecond is acceptable. SHM strategies need to be examined against time synchronization error prior to their use.

4. Limited computational capability

Some SHM applications utilize numerical operations that require large computational power and memory allocations. Methods requiring manipulation of large matrices, such as ERA, fall into this category. Because memory space on smart sensors is limited, these methods are impossible to implement on smart sensors, or the performance is limited (Chintalapudi et al., 2006; Nagayama et al., 2004).

5. Power

Power consumption is an important issue to civil infrastructure monitoring. As compared to other smart sensor applications (e.g., habitat monitoring), monitoring of structures may have easier access to power sources. Buildings are equipped with power outlets, and many long bridges have power that is used for light poles, etc. However, wiring power to numerous sensors takes a significant amount of time and increases installation cost, negating one of the important advantages of smart sensor. Also, sensor installation location is not always close to a power source on a structure. Available power may need to be transformed to an appropriate voltage and frequency. Some structures are without power. Therefore, power source cannot be assumed to be available. Smart sensors employing batteries are beneficial and oftentimes the only solution.

Once smart sensors are installed on civil infrastructure, their batteries may not be easily changed. Some nodes may be installed at places with low accessibility. Maintenance during which batteries on smart sensors can be replaced is not frequently scheduled. Power consumption for SHM applications is, however, in general larger than other wireless sensor applications, shortening battery life. So far, no battery-powered smart sensor systems for SHM has been implemented on a permanent basis. Much research effort is being devoted to resolve this issue; one promising approach is power harvesting.

2.5 Summary

Recent technological advance in smart sensor technology offers new opportunities in SHM for civil infrastructure. A dense array of smart sensors is expected to be a rich source of information for SHM. Attempts toward SHM using smart sensors, however, always encounter difficulties. Although there are reports on smart sensor usage in structural vibration measurement, none of them has resulted in a full-fledged SHM system. Issues in scalability, sensing capability, synchronization accuracy, etc. must still be addressed. This research realizes a framework for SHM using smart sensors by addressing these issues. The next chapter presents the SHM architecture employed in this report to address these difficulties.

Chapter 3

SHM ARCHITECTURE

This chapter discusses the architecture for an SHM framework that addresses many of the difficulties described in the previous chapter. Prior to architecture design, desirable characteristics of an SHM framework employing smart sensors are summarized. The smart sensor network system, smart sensor platform, middleware services, and damage detection algorithms to be employed in this report are then discussed in detail (Spencer & Nagayama, 2006).

3.1 Desirable characteristics of an SHM system employing smart sensors

Desirable characteristics for an SHM framework using smart sensors must first be established. Many researchers have been working on an SHM employing smart sensors by a variety of approaches. Some researchers utilize only the wireless communication capability, while others emphasize the use of the embedded microprocessor. Assumptions on the type of power source also vary widely. Desirable characteristics for an SHM strategy implemented on a network of smart sensors are specified herein and serve as guidelines for this research.

1. Scalable resource aware system

Scalability

A large number of sensors densely distributed over a region of interest are considered necessary to understand the complete state of a structure. Because damage to a structure is a local phenomenon, signals from sensors near damage are expected to contain more information on damage than those remotely located. The spatial variation of measurands can be accurately assessed only when sensors are distributed in a sufficiently dense manner. Dense instrumentation on structures such as the 3.9 km-long Akashi-Kaikyo Bridge and the 443 m-tall Sears Tower need a great number of smart sensor nodes. The inexpensive nature of the smart sensors potentially makes a large number of smart sensors economically feasible, while the wireless communication capability dramatically reduces installation costs. A network of smart sensors, however, is not scalable to a large number of sensors unless the limited resources in the network are respected and well-managed. For example, smart sensor networks that are intended simply to collect all of the measured data to a single base station suffer from insufficient bandwidth and are not scalable. Both the hardware and software should allow dense arrays of smart sensors.

Power awareness

Power available for smart sensor networks is typically limited, with most smart sensors being battery-operated. Power consumption needs to be managed so that the smart sensor's lifetime is prolonged. In particular, radio communication and flash memory access need to be planned well, as they are power-demanding tasks (see Table 2.1 on page 12). Within the network, nodes with more available power should be assigned more computation and communication tasks to extend the lifetime of the overall network. Effective power management should result in longer lifetimes for both individual nodes and the network.

Bandwidth awareness

The use of RF communication should be well-managed in order to limit network congestion. Excess communication increases data collision, and may even surpass the network's communication capacity. The higher the traffic load on a network, the more likely collisions will take place and the less reliable the communication will be. As a network, nodes with more available communication bandwidth should be assigned more relay tasks. To increase reliability and efficiency of communication, as well as to save power, data transferred needs to be kept moderate and bandwidth use should be optimized.

Memory awareness

Flash memory and RAM are among the limited resources on smart sensors. Moreover, accessing flash memory is power-intensive; read and write accesses should be limited. Variables needing frequent accesses should be kept on RAM instead of flash memory. RAM on a smart sensor, on the other hand, is a component which constantly consumes nonnegligible power; consequently, the current RAM size on smart sensors is relatively small and expected to increase slowly in the future. For example, Mica2 has only 4 kB of RAM, which allows storage of only about 1,000 data points in a single-precision floating point format. When an application requires processing of large data, such as Discrete Fourier Transform (DFT) of a long time history, Virtual Memory (VM) is one approach to address the problem of small RAM at the expense of flash memory usage and associated power consumption. Memory usage needs to be well-managed.

Computational power awareness

As compared to a microprocessor on a PC, the one on a smart sensor has limited speed and functionality. Such a microprocessor is in charge of multiple tasks, for example, data acquisition, flash memory access, RF control, numerical calculation, and task scheduling. Only a single task, or a few tasks if the microprocessor is designed specifically, can be executed at a time. The tasks on a microprocessor need to be well-organized with priorities, so that timing critical tasks, such as data acquisition, is not disrupted or delayed by other less-time-critical tasks.

Modal operation

Several modal operations for the microprocessor, such as sleep, watchdog, and awake, facilitate a resource-efficient operation. For example, following a preprogrammed schedule, nodes enter the watchdog mode to save power. When no significant event to observe is expected, the nodes go into the sleep mode. The smart sensors should be able to transition from one operation mode to another, depending on the tasks to be achieved.

2. Autonomous distributed embedded computing

Model-based data aggregation

The data measured at nodes should be processed locally so that a reduced amount of data needs to be sent throughout smart sensor networks. Data size reduction without data degradation should be sought. Algebraic operations such as averaging, numerical filtering, and resampling, are simple examples. More complex mathematical manipulation such as frequency analysis may better compress information. Data aggregation based on knowledge or insight on a structural system is expected to further condense measured data without compromising the structural information contained. Ideally, only necessary and sufficient information for the task is transmitted throughout networks.

Collaborative distributed data processing

Distributed data processing eliminates the problem of having a single point of failure and balances the power consumption among nodes. Also, distributed processing offers efficient computation, which can be much faster than centralized processing. Distributed processors, accompanied by cache memory and RAM, contribute to fast computation. Although resources on each node are limited, sensor networks as a whole possess appealing computational capabilities.

Autonomous initial configuration and maintenance

The network topology is desired to be dynamically and autonomously configurable. When smart sensors are physically installed, the sensor nodes need to construct a network and configure the topology. Because node loss is likely to take place, network configuration should be adjusted autonomously so that loss of a single node does not take the system down. Autonomous reconfiguration also allows balancing power consumption among sensor nodes by switching relay nodes on multihop communication paths.

Individual nodes should be reconfigurable through the network as well. For the initial set up of a smart sensor network, a large number of nodes need to be programmed as well as physically mounted. In the long term, the nodes may need to be reprogrammed to implement different tasks or to reschedule the tasks. Smart sensors are desired to be programmable through the network, because manual reprogramming of thousands of nodes is too time consuming and error prone.

3. Fault tolerant system

Message packet loss tolerance

RF communication inherently has data loss. Resending lost data packets can increase transmission reliability at the price of increased communication demand and power consumption. Algorithms tolerant of data loss are appealing from both power and bandwidth perspectives.

Node failure tolerance

During the life of a smart sensor node, functionalities of the node may become impaired. Power depletion stops all of the functionalities of a smart sensor. Wireless link disconnection impairs only the communication capability. The network should be tolerant of these node losses by reconfiguration of networks, while application algorithms allowing the loss of sensing signals from a few nodes are desirable.

Byzantine error tolerance

Even when data is acquired, the data can be faulty and misleading. For example, when a node with an accelerometer comes unglued, the node acquires faulty acceleration data from the sensor. As another example, RF interference could induce errors in received data. Low battery power may also result in random errors in the radio, CPU, and/or sensor. A system to address this Byzantine error problem is desirable.

4. Desirable algorithmic characteristics specific to SHM for civil infrastructure

Multiscale information

SHM techniques are needed that can employ measured information at multiple scales. Different types of sensors measure different physical quantities, each of which has its own sensitivity to certain structural conditions. For example, acceleration and strain are among the most important physical quantities to judge the health of a structure. While acceleration measurements are essentially global responses of a structure, structural strain provides an important indicator of local structural behavior. By using such multiscale sensor information, structural condition is expected to be assessed more accurately.

Collaboration in local sensor communities

Smart sensors, densely distributed over structures, are expected to communicate with each other, at least in a local sensor community, to make efficient and effective use of measured data. Closely located nodes are anticipated to give highly correlated measurements, which might be fused without significant loss of information, although the definition of closeness depends on each measurand. Data processing of a spatially sampled measurand may reveal important information, in a similar way to data processing of a measurand sampled in the time domain. For example, estimation of a mode shape and its spatial gradient, which is claimed to change according to the health of a structure (Farrar et al., 1994; Nagayama et al., 2005; Pandey et al., 1991; West, 1984), needs spatial information. Independent data processing without data sharing among nodes cannot utilize such spatial information. On the other hand, collaboration of all of the smart sensors in the whole network is neither practical nor desirable due to substantial communication and power requirements. Collaboration in local communities, which are spatially large and dense enough to capture local structural condition, is thus essential for SHM using a dense array of smart sensors.

Redundancy

Because measurement error, inaccurate modeling, numerical error, etc. introduce uncertainty in SHM results, SHM algorithms on smart sensor networks should possess redundancy. For example, more than one set of smart sensors monitor a given element and make their own judgments on the damage existence; the judgments are then compared with each other to examine their consistency. Redundancy is an effective means to deal with the uncertainty, which is problematic for most SHM algorithms.

Multiple functions

Smart sensor networks should preferably achieve multiple tasks, for example, continuous SHM of a structure, baseline measurement for SHM, monitoring of rare events such as earthquakes, and sensor calibration. Inclusion of other tasks such as traffic monitoring and local weather monitoring further enhances the value of smart sensor systems and makes introduction of smart sensor systems more attractive from a cost/ benefit perspective.

5. Desirable platform characteristics specific to SHM of civil infrastructure

Appropriate sensor availability

Appropriate sensors need to be available for smart sensors. Acceleration and strain are among the most important measurands for SHM, while velocity, displacement, temperature, humidity, wind velocity, and wind direction as well are sometimes measured in full-scale structural monitoring (Ko & Ni, 2005; Wong, 2004). Sensors for these measurands are needed on smart sensor platforms. One of the most commonly adopted sensors on smart sensor prototypes is an accelerometer. The applicability of accelerometers on smart sensors to civil engineering applications is, however, not necessarily clear. Strain sensor adoption to smart sensors is not common (Arms et al., 2004; Lynch & Loh, 2006; Nagayama et al., 2004). Sensors being capable of accurately capturing structural behavior and environmental conditions need to be developed or customized under the constraint of limited resources of smart sensors, especially battery power.

Sufficient RAM, flash memory and RF bandwidth

Because SHM for civil infrastructure measures dynamic vibrations of large structures at high sampling frequencies, the amount of data to be saved, processed, and transmitted is large even after local data processing. The sampling frequency of dynamic vibration measurements is often higher than 100 Hz. RAM size, flash memory size, and radio throughput are desirably large, though power consumption still needs to remain moderate.

Environmentally hardened

Because smart sensors placed on structures are often subjected to severe environmental conditions (e.g., wind, rain, sunshine, extreme temperature, and vibration), the sensors need to be rugged. Packaging is one solution. Without ruggedness, severe environmental conditions may result in faulty data or node loss.

Open source OS and interface

The operating system (OS) and the interface should be open to the public. Tasks are ideally optimized to make efficient use of the limited resources. Services offered by the OS may not give the best solution for a specific task. Algorithms and middleware services to be implemented may work more efficiently if the OS is customized. Certain algorithms or middleware services may be implementable only if the OS is customized. In such cases, users should have access to the source of the OS. Also, open source OS and interfaces allow user communities to participate in improving OS and software. In particular, at this early stage of smart sensor network research, contributions from broad communities are important to improve the technology. Open source OS and interfaces will advance smart sensor network technology and its application to SHM.

3.2 SHM system architecture

An SHM system architecture to address many of the difficulties explained in the previous chapter and the desirable characteristics is proposed herein. Network system architecture, smart sensor platform, middleware services, and algorithms to be employed in this report are described.

3.2.1 Network system architecture

A homogeneous configuration with a single type of node is employed instead of the tiered system approaches employing powerful upper-level nodes and less powerful lower-level nodes. Note that even though only one type of node is deployed over a structure, a PC may be needed in this architecture as an interface to users. Usage of only one type of node allows for development of a simple and robust sensor network system, as explained later in this section.

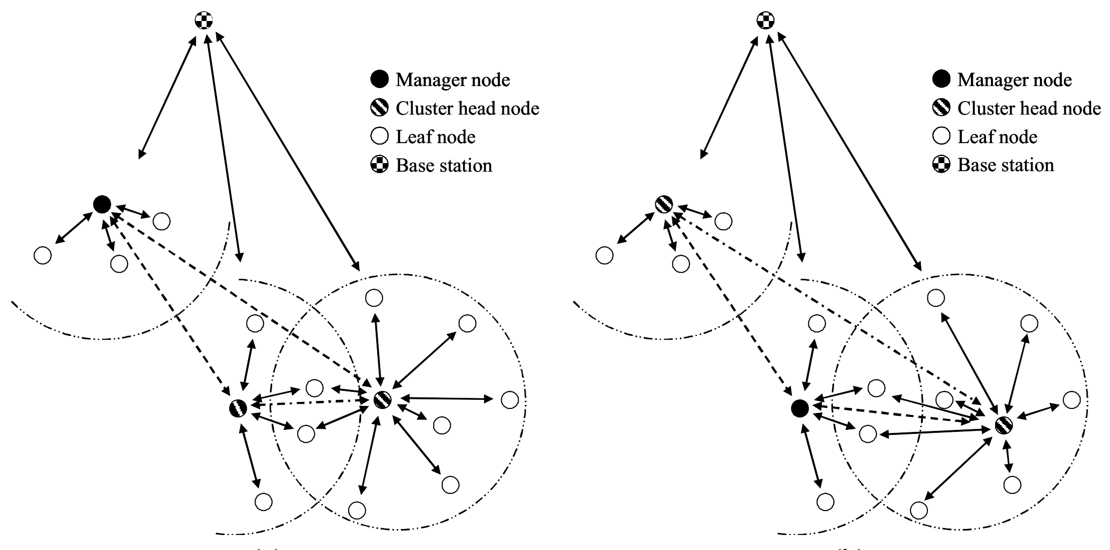


Figure 3.1. SHM system architecture: Different roles are assigned to nodes in (a) and (b).

In terms of functionality, smart sensor nodes in the proposed system are differentiated as follows: the base station, the manager node, cluster heads, and leaf nodes. All of the sensors deployed on a structure, in principle, work as leaf nodes. Leaf nodes receive commands from the other nodes and perform preprogrammed tasks such as sensing, data processing, and acknowledgment. A node in a local sensor community is assigned as a cluster head and coordinates most of the communication and data processing in the community. In addition to tasks inside the community, the cluster head communicates with the cluster heads of the neighboring communities to exchange information. One of the cluster heads also functions as the manager node. When intracluster RF communication spans multiple clusters, the manager deals with time sharing among clusters to avoid RF interference. The manager also exchanges packets with the leaf nodes to manage operations in which all of the leaf nodes participate; sensing that is triggered by the manager sensor is an example. The base station node is the gateway between smart sensor networks and the PC. The PC with a user interface sends commands and parameters to smart sensor networks via the base station. The PC also receives data and calculation results from the base station. While the base station can communicate with any node in direct communication range, most communication involving the base station is routed through the manager or cluster heads with the exception being transmission of a large amount of data or calculation results from leaf nodes to the PC for debugging purposes. Thus, smart sensors nodes are functionally differentiated into four categories (see Figure 3.1).

This system architecture can be compared with tiered networks to clarify its characteristics. A tiered network assumes ample hardware resources at the upper level nodes. By reducing the constraints on hardware (i.e., power source, RAM space, flash memory space, radio bandwidth, etc.), a tiered network aims to implement more functionality on the network easily. However, such an assumption is not necessarily the case. When smart sensors are installed on a long suspension bridge, there needs to be many powerful upper-tier nodes connected to power sources. Preparing power sources for these upper-tier nodes spatially distributed over a long structure is expensive, negating one of the appealing features of smart sensors. The proposed system architecture using a homogeneous configuration does not assume such ample hardware resource availability.

Another advantage of a homogenous configuration is its robustness to node failure. All of the smart sensors distributed over civil infrastructure in this architecture have the same hardware configuration; thus, their roles can be reassigned in the case when some nodes stop working due to battery exhaustion, OS failure, etc. Cluster heads and manager sensors can be chosen from the survivors. Robustness to node failure is an important feature for smart sensor networks, because node failure is an easily anticipated event.

Reassignment of roles can also be performed based on available hardware resources. The roles of smart sensor nodes can be switched as shown in Figure 3.1. When the battery voltage of a cluster head becomes lower than a threshold, another node in the neighborhood with ample battery power can take over the role of the cluster head. If necessary, only a part of the cluster head's tasks can be reassigned to neighboring nodes. In this manner, resources distributed over a large number of smart sensors can be efficiently utilized.

3.2.2 Smart sensor platform

The Imote2 is chosen as the smart sensor platform for an SHM system. The Imote2 has much larger RAM space, larger flash memory size, and a faster on-board processor as compared to other smart sensor platforms as shown on Table 2.1 on page 12. This hardware specification is considered as being able to provide the necessary data processing required by DCS for SHM. The base station Imote2 can communicate through serial ports with the PC connected to a USB cable and the programming board.

The sensor board employed in this research has an LIS3L02DQ triaxial digital accelerometer (STMicroelectronics, 2007). The accelerometer employs a sigma-delta ADC (Stewart & Pfann, 1998) to obtain the digital signals. The ADCs produce bitstreams representing the acceleration signals. Sinc^3 filters are then applied to the bitstreams to reconstruct the signal at the specified sampling rate. These reconstruction filters have a low cutoff frequency so that the output signals are decimated without the signal being aliased. The user can select a decimation factor from 8, 32, 64, and 128. The cutoff frequency of the reconstruction filter and the sampling rate of the signal is fixed by the decimation factor. The Imote2 processor board receives these digital signals through the I2C or SPI ports on its basic I/O connector.

TinyOS is employed as the operating systems on smart sensors. This operating system fits in small memory footprint and is suited for smart sensors with limited resources. TinyOS has a large user community and many successful smart sensor applications. However, from a civil engineering perspective, TinyOS may pose limitations on the SHM system's functionalities. TinyOS does not support real-time operations. In other words, the operating system has only two types of threads of execution: tasks and hardware event handlers, leaving users with only a little control to assign priority to commands; execution timing cannot be arbitrary controlled. This feature of TinyOS needs to be well-considered when designing a system

3.2.3 Middleware services

Middleware services are developed to provide functionalities that TinyOS does not offer but that are needed by SHM applications. The middleware services considered herein are model-based data aggregation, reliable communication, and synchronized sensing. Model-based data aggregation utilizes application-specific knowledge to efficiently collect information from measured data in a network. Model-based data aggregation can save communication resources and contributes to scalability of a smart sensor network. The reliable communication service enables communication without loss of information by sending packets repeatedly. This service addresses the problem of lossy communication. Synchronized sensing utilizes a time synchronization service and obtains synchronized signals from a network of smart sensors. Synchronized clocks on smart sensors in a network does not mean measured signals are synchronized, because smart sensors cannot necessarily control sensing task timing precisely based on their clocks. This service contributes to accurate measurement in terms of sensing timing. These middleware services are realized on the smart sensor platform.

3.2.4 Damage detection algorithm

The damage detection algorithm employed in this work is an extension of the Distributed Computing Strategy (DCS) for SHM proposed by Gao (2005). Smart sensors form local sensor communities. Local sensor communities measure acceleration responses of a structure and perform modal analysis. Locally determined modal parameters are then utilized to construct a portion of the flexibility matrix of the structure. Changes in the flexibility matrix before and after damage are then analyzed with Singular Value Decomposition (SVD) to estimate the Damage Locating Vectors (DLVs). DLVs are applied to the numerical model of the structure as input static force, and elements with small stress are identified as potentially damaged elements. This damage detection is performed in each local sensor community. Cluster heads communicate with each other in order to exchange information about damaged elements. The details of this approach are explained later in Chapter 6. By distributing and coordinating data processing, the DCS for SHM and the proposed extension offers a solution for SHM system employing a dense array of smart sensors.

3.3 Summary

This chapter describes desirable characteristics for smart sensor SHM systems and the SHM architecture employed in this report. The difficulties, desirable characteristics, and approaches in this report are summarized in Tables 3.1 and 3.2. The next chapter demonstrates customizability of sensor boards, which leads to availability of appropriate sensors.

Table 3.1. *Difficulties in SHM Using Smart Sensors and Approaches in This Report*

Difficulty	Approach
Sensor hardware	• The Imote2 sensor board, which has a triaxial accelerometer, is employed. The performance of the accelerometer is compared with reference sensors. • Sensor board customizability is demonstrated.
Data aggregation	• Model-based data aggregation is proposed. • DCS for SHM allows distributed data processing, eliminating the need for centrally collecting a large amount of data.
Time synchronization	• Synchronized sensing middleware service is proposed.
Limited computational capability	• Computational tasks are distributed by employing model-based data aggregation and DCS for SHM. • The Imote2 has a relatively fast microprocessor as well as large RAM.
Power	• Power consumption is kept moderate by employing DCS for SHM and model-based data aggregation. • CPU clock frequency is switched, depending on tasks to perform. • Power harvesting is not addressed in this research.

Table 3.2. *Desirable Characteristics and Approaches in This Report*

Desirable characteristic	Approach
Scalable resource aware system	
Scalability	• DCS for SHM and model-based data aggregation offers scalability.
Power awareness	• RF and flash memory usage are kept moderate through DCS for SHM and model-based data aggregation, as well as implementation ingenuity. • The homogeneous hardware configuration of the network potentially offers reassignment of roles based on available power.
Bandwidth awareness	• RF communication is managed by manager node and cluster heads. • Reliable communication protocols, which wait for longer intervals in a congested communication network, are developed.
Memory awareness	• SHM system is designed so that requests for memory usage do not exceed the currently available memory size of the Imote2, that is 256 kB.
Computational power awareness	• Computational tasks are distributed by employing DCS for SHM and model-based data aggregation. • The frequency scalable processor on the Imote2 is utilized to make efficient use of computational power.
Modal operation	• Smart sensors get ready to enter sleep mode. • TinyOS that supports modal operation is employed.
Autonomous distributed embedded computing	
Model-based data aggregation	• Model-based data aggregation is proposed and implemented.
Collaborative distributed data processing	• DCS for SHM and model-based data aggregation realize collaborative, distributed data processing.

Table 3.2. *Desirable Characteristics and Approaches in This Report (Continued)*

Desirable characteristic	Approach
Autonomous initial configuration and maintenance	• Not realized in this report.
Fault tolerant system	
Message packet loss tolerance	• Reliable communication protocols are proposed.
Node failure tolerance	• The homogeneous hardware configuration of the network potentially addresses the node failure.
Byzantine error tolerance	• Not realized in this report.
Desirable algorithmic characteristics specific to SHM for civil infrastructure	
Multiscale information	• Not realized in this report.
Collaboration in local sensor communities	• DCS for SHM and model-based data aggregation offers this collaboration.
Redundancy	• DCS for SHM gives redundancy through overlapping clusters.
Multiple functions	• Baseline measurement and continuous SHM of a structure are implemented on smart sensors. • Rare event monitoring, traffic monitoring, local weather monitoring, etc. are not realized.

Table 3.2. *Desirable Characteristics and Approaches in This Report (Continued)*

Desirable characteristic	Approach
Desirable platform characteristics specific to SHM for civil infrastructure	
Appropriate sensor availability	• The Imote2 sensor board, which has a triaxial accelerometer, is employed. The performance of the accelerometer is compared with reference sensors. • Sensor board customizability is demonstrated.
Sufficient RAM, flash memory, and RF bandwidth	• The Imote2 has relatively large RAM and flash memory. • The Imote2 has a ZigBee-compliant RF component. The throughput is larger than RF components on other smart smart sensor platforms such as Mica2.
Environmentally hardened	• Not realized in this report.
Open-source OS and interface	• TinyOS is employed.

Chapter 4

SENSOR BOARD CUSTOMIZATION

The lack of appropriate sensors and sensing components has hindered progress for SHM applications with smart sensors. In this chapter, the customizability of sensor boards is demonstrated. In particular, a strain sensor board and an AA-filter board will be designed, constructed, and tested. Here, the Berkeley Mote, instead of the Imote2, is chosen as the smart sensor platform because of its availability during the early stages of this research. The customization procedure for the Imote2 is the same as that for the Berkeley Mote. Also, the Berkeley Mote is designed for generic applications, while the Imote2 has been developed for more data intensive applications. Development of a Berkeley Mote sensor board demonstrates that a general-purpose sensor board can be customized for SHM applications.

The Berkeley Mote was designed for generic applications; available sensors have not been optimized for use in civil infrastructure applications. While acceleration measurements are essential to obtain global responses of a structure, structural strains provide an important indicator of local structural behavior. Studer and Peters (2004) demonstrated that multiscale sensing yields better results than single-scale measurements for damage identification. However, commercially available sensor boards only have accelerometers for dynamic structural response measurement, and their applicability to civil infrastructure is limited (Ruiz-Sandoval et al., 2006).

Although a great variety of strain sensors exist, strain sensor boards have yet to be developed for the Berkeley Mote platform. The search for, selection of, and testing of an appropriate strain sensor tailored to the Mote platform are presented herein. An AA filter board to properly condition the signal from the strain sensor board to the Mica2's ADC is also designed, constructed, and tested. Experimental results demonstrate that the strain sensor board has good resolution, and that the output is comparable with conventional wired strain sensors.

A limitation in the Mica2's hardware was identified during the experimental verification - a systemic fluctuation was observed in the acquired data during reading/ writing of data by the Mica2's microprocessor. This problem is addressed through careful design of the analog circuit and subsequent digital signal processing.

4.1 Strain sensor board development

Many different types of strain sensors exist that use diverse methods for measurement, such as mechanical, optical, or electrical means. Mechanical strain sensors, for example a slide caliper, are simple but generally have poor resolution. A device with levers to amplify strain to readable values can be devised, but it is prohibitively large. Optical sensors are typically costly and too delicate to be densely deployed on civil infrastructure. Electrical sensors, which include the piezoelectric sensor, semiconductor

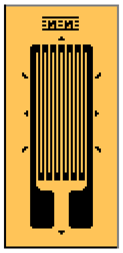


Figure 4.1. Foil strain gage.

strain gage, and the widely used foil strain gage (see Figure 4.1), have high resolution and are small, sturdy, and inexpensive. Thus, they are good candidates for developing a wireless strain sensor. Several desirable characteristics for a smart wireless strain sensors for civil infrastructure applications are presented in the following paragraphs along with a discussion of the associated impact on the design.

First, low-frequency responses (e.g., <1 Hz) typically found in tall or long civil infrastructure need to be measured; therefore, sensors with DC capability are preferable. Polyvinylidene fluoride (FVDF) film sensors, one type of piezoelectric sensor, are appealing as they are rugged, inexpensive, and have low power requirements. However, their sensitivity in the low frequency range is poor; researchers are working to resolve this difficulty (Satpathi et al., 1999), but it is still problematic. As for semiconductor strain gages, though large sensitivity to strain is an advantage, its sensitivity to variations in temperature and tendency to drift are nontrivial disadvantages. In contrast, foil strain gages (see Figure 4.1) have a wide frequency range and possess a DC capability. The gage is a metallic resistor whose resistance varies almost linearly with its strain. A gage bonded to an object deforms with the object's surface and yields resistance change. A circuit with a Wheatstone bridge and amplifier converts the resistance change to a readable voltage change. For this project, foil strain gages were chosen.

Second, low power consumption is an important characteristic as the wireless sensor network usually operates on local battery power. For example, one prototype of the Berkeley Mote can operate for up to a year on a single battery while in the power-down mode. The power-down mode shuts off everything except for a watchdog and the asynchronous interrupt logic necessary for wake up. However, it operates for only 30 hours at peak load (Hill et al., 2000). The power consumption in a strain sensor's Wheatstone bridge is inversely proportional to the resistance. Thus, a high resistance strain gage (e.g., 4,500 Ω) is chosen as opposed to the widely used 120/350 Ω gages, in an effort to moderate power consumption.

The measurement range selected is 1 to 2,000 με. The lower limit is set based on the resolution of a commercial wireless strain sensor product, the SG-Link Wireless Strain Gauge System (MicroStrain, Inc., 2007). The upper limit is set based on the yielding strain of steel. To achieve a wide measurement range despite the Mica2's 10-bit ADC restriction, a variable gain amplifier was implemented.

Finally, a target noise level is sought that is equal to the target resolution, 1με. Significant high frequency noise was found to be present in the strain sensor board. The low-pass filter described in the next section was employed to remove the high-frequency

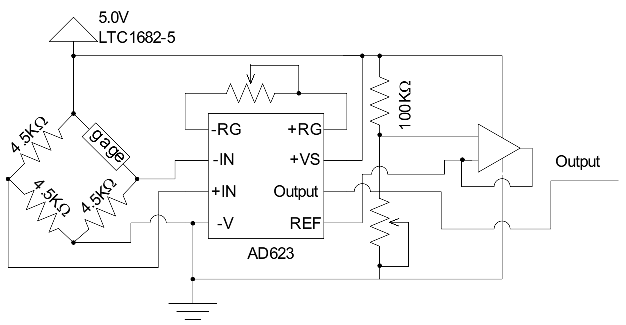


Figure 4.2. Wireless strain sensor circuit schematic.

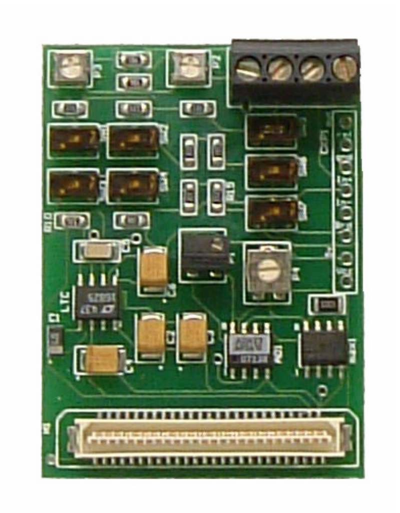


Figure 4.3. Strain sensor board.

noise. This filter also reduces the problem of aliasing. To have a larger signal-to-noise ratio and to mitigate problems with supply power fluctuation in the Mica2, a voltage doubler/regulator, LTC1682-5 (Linear Technology, 2007), was added, which provides a constant 5 V excitation for the strain bridge. Note that an amplifier with low noise, the AD623 (Analog Devices, Inc., 2007), was selected for this circuit (see Figure 4.2).

The strain sensor board was fabricated as shown in Figure 4.3 (Screaming Circuit, 2007; PCBExpress, 2007). The terminal block in the top right corner of the strain sensor board is wired to the strain gage's terminals. Using appropriate wiring and the switch settings on the board, the strain sensor board can be configured both as a quarter-bridge circuit and a half-bridge circuit. Switches on the strain sensor board allow any of the three different gages (i.e., 120/350/4,500 Ω strain gages) to be employed.

4.2 AA filter board development

Because the Berkeley Mote platform does not have antialiasing (AA) filters on the processor/radio unit, a new AA filter board needs to be developed to address the aliasing problem. When a continuous signal is sampled at a sampling frequency, f_s, any signal

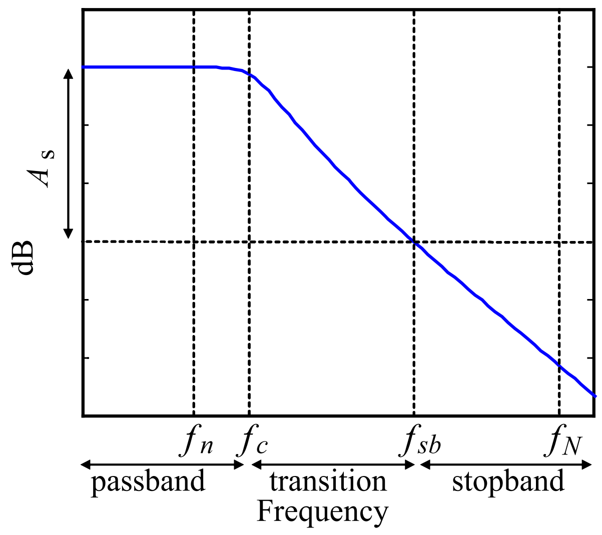


Figure 4.4. AA filter design parameters.

component whose frequency is higher than the Nyquist frequency, $f_N = f_s/2$, is folded back to the frequency range from 0 to f_N, i.e., the signal is aliased. As a result of this sampling phenomenon, signals in the frequency range above f_N are superimposed onto the original signal components in the baseband frequency range after the sampling. Once the signal is contaminated with aliasing components, it cannot be corrected. Therefore, the high-frequency components above the Nyquist frequency need to be eliminated prior to the sampling process.

Ideally, a Linear Time Invariant (LTI) analog circuit, whose gain is unity over the passband range with linear phase response and then attenuates quickly to zero is desirable. In practice, LTI circuits close to the ideal circuit are employed as AA filters (see Figure 4.4). To completely eliminate aliasing in a digital signal, the following relation needs to hold

$$f_n < f_c < f_{sb} < f_N \tag{4.1}$$

where f_n is the highest frequency of signal components to be analyzed; f_c is filter's passband cutoff frequency; and f_{sb} is the stopband cutoff frequency. The stopband attenuation, A_s, is determined so that any signal in the stopband is smaller than the resolution of the ADC connected to the filter. Other requirements for AA filters include small-gain variance in the passband; and linear phase over the passband, which keeps the signals undistorted in the time domain.

There are many variations in LTI circuits for AA filters. Though an arbitrary LTI system, which satisfies the above mentioned requirement, works as an AA filter, several filter types have been proposed and used for their specific characteristics. The Butterworth filter, which is also called the "maximally flat magnitude" filter, has a frequency response that is as flat as mathematically possible in the passband. The Bessel filter has the maximally linear phase response. The elliptic filter has an equiripple magnitude response in both the passband and stopband, minimizing the maximum error in both bands. The order of the filter design, or number of poles, is also a filter parameter. High-order filters, in general, have more freedom in filter design, giving better magnitude characteristics at the expense of design difficulty (Skolnick & Levine, 1997) and larger time lag; the slope of a filter's transfer function phase is the time lag.

If an 8-pole, elliptic-type AA filter along with 16-bit ADC is employed as in many applications in civil engineering, the attenuation, A_s is about -96 dB ($20\log 2^{16} = 96$), and f_{sb} is 1.28 times f_c . Therefore, the sampling frequency is set about 2.56 times the highest frequency signal of interest. The combination of another type of AA filter and ADC with different resolution gives different values for the attenuation and the stopband cutoff frequency.

A 4-pole, Butterworth low-pass filter with a cutoff frequency of 50 Hz was selected as the AA filter for the Berkeley Mote platform. The filter order and cutoff frequency were set in order to investigate concurrently the 'Tadeo' acceleration sensor board developed by Ruiz-Sandoval (2004) and the strain sensor board characteristics over a wide frequency range (i.e., 0-50 Hz), and to limit the associated time lag. A small time lag is important for applications with real-time response requirements, such as structural control and triggering smart sensor tasks based on the measurement; high filter order and a low cutoff frequency result in large deterministic time lags. From the AA perspective, this 4-pole filter cannot completely eliminate aliasing for Mica2's 10-bit ADC when sampled at 100 or 250 Hz. f_{sb} for a 10-bit ADC is around 230 Hz. The noise is, however, unlikely to have an amplitude as large as the ADC's input range. Small noise components below f_{sb} and all of the components above f_{sb} are completely eliminated by this filter.

There are three major designs to implement the AA filter as an electrical circuit: (a) passive filter, (b) active filter, and (c) switched capacitor filter designs (National Semiconductor, 1991). A passive filter is made up of passive components without amplifiers, i.e., resistors, capacitors, and inductors. Though having a small noise level, being free from power supply, and having the capability to deal with a large voltage and current are advantages of this filter design, inflexibility in the design of the filter is detrimental. Active filter designs employ operational amplifiers, eliminating the need for the hard-to-handle inductors, and have more flexibility in design. The noise level is higher than that of the passive design, but it is still moderate. Switched-capacitor filters are clocked, sampled-data systems widely available in monolithic form. The cutoff frequency is typically set by an external clock frequency. This filter has design flexibility, a variable cutoff frequency, and insensitivity to temperature change. Poor DC precision and high noise level, however, need to be addressed by additional external circuits, according to application requirements. Frequency components higher than half the clock frequency cannot be eliminated by this filter; they need to be eliminated with passive or active filters. In this research, an active design was employed because of its design flexibility, moderate noise level, and good DC precision.

A 4-pole Butterworth filter was realized by the Sallen-Key active filter design (Sallen & Key, 1955), as shown in Figure 4.5. Rail-to-rail input/output amplifiers, MAX4132 (Maxim Integrated Products, Inc., 2007), give the filter a rail-to-rail input/output property, which results in the efficient use of the input/output voltage range. The cutoff frequency

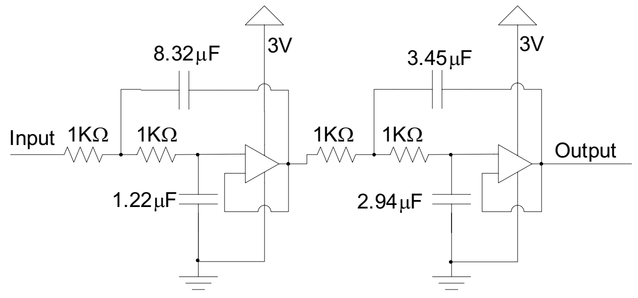


Figure 4.5. Sallen-Key fourth-order active low-pass filter.

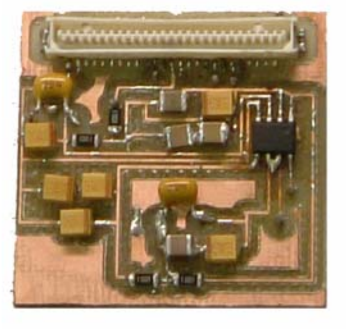


Figure 4.6. Four-pole AA filter board.

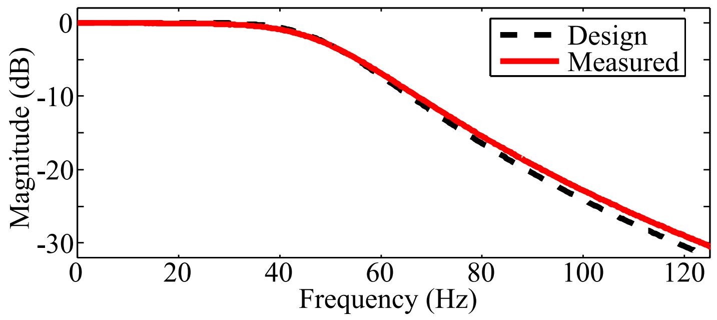


Figure 4.7. Four-pole AA filter board transfer function.

was set as 50 Hz by appropriate choice of resistors and capacitors. Note that filters with other cutoff frequencies can be easily constructed by changing resistors and capacitors.

The final printed circuit board (see Figure 4.6) was manufactured at the Electronics Services Shop (ESS, 2007) based on the cascaded two second-order filter design. The filter's transfer function was verified as shown in Figure 4.7. The difference between the design value and the measured value of the transfer function is within the expected range of error due to finite precision of the capacitors on the board.

Moreover, another AA filter, an 8-pole Butterworth low-pass filter with a cutoff frequency of 25 Hz, was designed and fabricated (PCBExpress, 2007; Screaming Circuit, 2007). This filter board has a stopband cutoff frequency, f_{sb}, of about 60 Hz. Aliasing is completely eliminated when sampled at 120 Hz or at a higher frequency, at the expense of

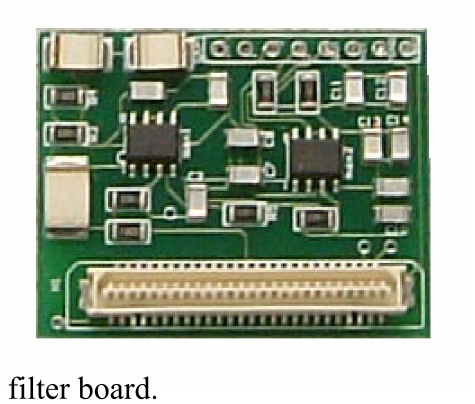


Figure 4.8. Eight-pole AA filter board.

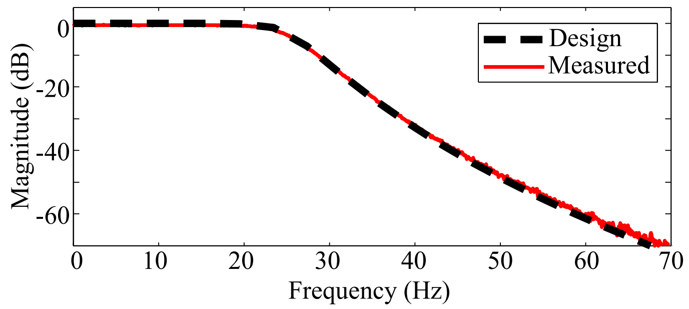


Figure 4.9. Eight-pole AA filter board transfer function.

a larger deterministic time lag. This filter is well-suited for measurements in which no signal of interest is above 25 Hz and the time lag of the filter, 34 ms, is acceptable. (see Figures 4.8 and 4.9)

4.3 Experimental verification of strain sensor

The strain sensor/AA filter boards must be calibrated before use. These boards are stacked on the Mica2, and shunt calibration is conducted to determine the sensitivity of the strain sensor. Shunt calibration simulates a resistance change in the strain gage by shunting the Wheatstone bridge with a known resistor. The output can then be calibrated. The data acquired by the Mica2 is first stored on on-board flash memory and then wirelessly transmitted to the base station (Mechitov et al., 2004). For convenience of calibration, the strain sensor board is equipped with four switches to shunt the bridge with different resistors that are wired in parallel with the strain gage. The 4-pole AA filter board is employed for this calibration as well as the experiments described subsequently. The sensitivity and offset of the strain sensor board is thus calibrated.

The sensor noise level must also be characterized. Based on the RMS of the measured signal, the resolution of the strain gage is estimated to be 3.0 $\mu\varepsilon$, which is slightly larger than the target noise level. By using a more precise amplifier, as well as electromagnetic shielding, further reduction in the noise level of the analog circuit is considered possible. In addition, the flash memory writing process is a noise source. The writing process needs a relatively large current and destabilizes the on-board power supply. Analog-to-digital

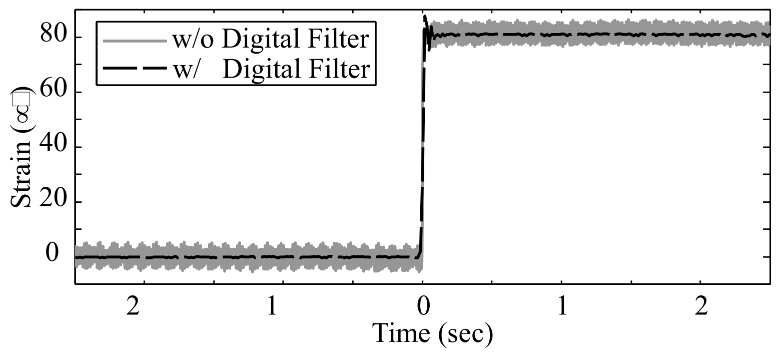


Figure 4.10. Shunt calibration.

conversion was noticeably affected by this process, introducing peak noise corresponding to flash memory access. This phenomenon needs to be suppressed.

Rather than seeking a hardware solution to reduce the noise level, the on-board microprocessor was exploited to yield a better resolution of the strain sensor. A frequency domain analysis indicates that the noise of the signal consists mainly of a 50 Hz component. The natural frequencies of the experimental structure model, described in detail in the following paragraph, are estimated to be smaller than 5 Hz. Consequently, a 12-pole Butterworth digital filter with a cutoff frequency of 25 Hz was employed to eliminate the 50 Hz noise. Because the Mica2 does not possess sufficient computational capability to apply the digital filter on-the-fly, the acceleration record is first stored in flash memory and subsequently digitally processed. Flash memory access was programmed to periodically take place at a frequency higher than the digital filter cutoff frequency so that noise resulting from flash memory access is also suppressed by the digital filter. In addition to simply applying a digital filter to the measurement data, downsampling is embedded to efficiently utilize available memory. Because the filtered signal does not have high-frequency components, the signal can be resampled at a lower sampling frequency without aliasing. The digital filter is applied to the original data sampled at 250 Hz, and then every fourth sample is stored on the flash memory. The resultant sampling frequency is, therefore, 62.5 Hz. The shunt calibration procedure is repeated with this digital filter (see Figure 4.10). The noise level of this strain sensor was, thus, reduced to 0.2 $\mu\varepsilon$.

The accuracy of the strain sensor/AA filter boards is experimentally verified using a three-story scale model building (see Figure 4.11). A strain gage was installed on the first-story wall of the structural model and connected to the strain sensor board on the Mica2. The acquired data was transmitted to the base station using wireless communication. A reference strain gage was also attached on this structure, and the gage was wired to a conventional strain measurement system. The outputs of the wireless strain sensor and the reference strain sensor were compared for the case where the structure was responding under free vibration. Because the strain sensor board has an adjustable amplifier, in this experiment the gain was adjusted to provide a large signal-to-noise ratio. Through these analog circuit considerations and digital signal processing, the test on the three-story scale

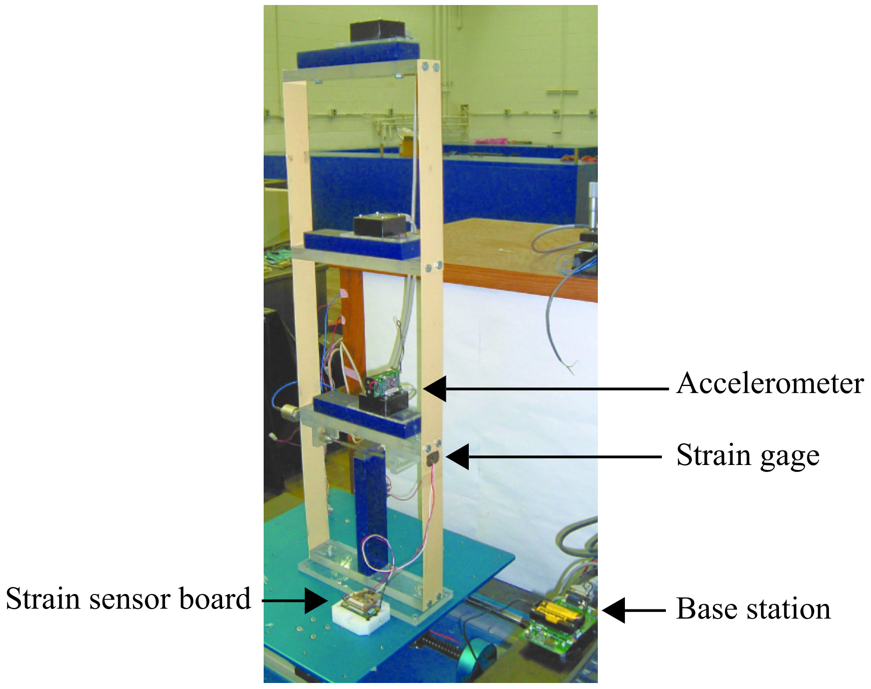


Figure 4.11. Experimental setup (a three-story building model and smart sensors).

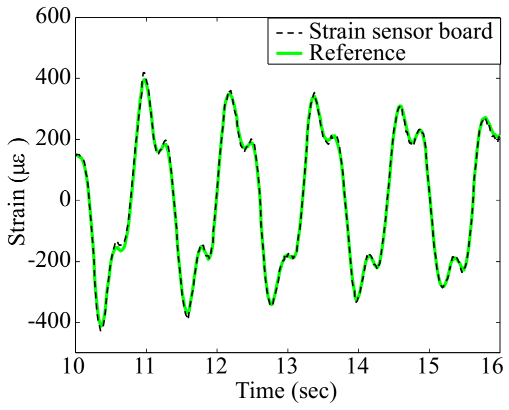


Figure 4.12. Strain sensor readings.

model showed that the wireless strain sensor has a good resolution and that the output is comparable to conventional wired strain sensors (see Figure 4.12).

4.4 Summary

The customizability of sensor boards was demonstrated using the Berkeley Mote platform. While strain is one of important physical quantities for SHM applications, strain measurement using smart sensors has been rare. No strain sensor board was available for the Berkeley Mote. Strain sensor and AA filter boards were developed and their performance was experimentally validated. Sensor board customization procedures for other smart sensor platforms are considered similar to those demonstrated in this chapter.

As demonstrated, physical quantities needed in SHM applications can be measured by customized sensor boards. These physical quantities are processed and transferred by middleware services and algorithms. In the next chapter, middlewere services will be developed.

Chapter 5

MIDDLEWARE SERVICES

In this chapter, middleware services for smart sensors are studied and realized. Among the middleware services are data aggregation, reliable communication, and synchronized sensing (Nagayama et al., 2007). These middleware services are implemented on the Imote2 running TinyOS, while ideas behind these services are generally applicable to other smart sensor platforms.

5.1 Data aggregation

The amount of data transferred in SHM applications is considerable. Long vibration records are acquired from densely distributed smart sensors. If they are collected at a single sink node using multihop communication, communication time easily exceeds the time necessary for any other smart sensor task. The amount of data utilized in SHM applications is first estimated. Distributed estimation of correlation functions is then proposed as model-based data aggregation (Nagayama et al., 2006b, 2007; Spencer & Nagayama, 2006). This data aggregation method is scalable to networks of large numbers of smart sensors. Implementation issues are then discussed.

5.1.1 Estimate on data amount in SHM applications

To capture the vibration response of civil infrastructure, sensors need to acquire data at an appropriate sampling rate for a sufficient period of time at various locations. Analysis methods are among the factors in determining these parameters. While there are many analysis methods for SHM applications, most output-only-measurement approaches for civil infrastructure utilize Power Spectral Density (PSD) or Cross Spectral Density (CSD) estimation. The amount of data involved in spectral density estimation for civil infrastructure, thus, can be estimated.

The number of data points used in spectral density estimation is determined by the number of FFT data points and the number of averages. Spectral density estimation is performed by averaging the outcomes of FFT analyses. Though any power of two theoretically works as the number of FFT data points, 1,024 or 2,048 points are most often used and give reasonable results. The error in the spectral density estimation is inversely proportional to the number of averages. A larger number of averages is, therefore, advantageous. The drawback of having a large number of averages is the associated large amount of data and long measurement time. If the measurement time is too long, environmental conditions such as wind velocity and temperature may change during measurement; averaging signals measured when a structure is behaving in a transient manner is not desirable. The number of averages in spectral density estimation practically ranges from 10 to 20. If 50 percent overlap is employed in the spectral density estimation with 2,048 FFT data points, 20 time number of averages needs 21,504 data points.

There are other factors to be considered when the number of data points is determined. One of them is the sampling frequency. The sampling frequency typically ranges from 100 to 256 Hz for civil infrastructure. These sampling frequencies are high enough considering the dominant modes of bridges, buildings, and towers. The frequencies of dominants modes are usually lower than 10 Hz. The reasons why the sampling frequency is much higher than the minimum frequency to capture vibrations below 10 Hz include that higher modes may also be influential to the response, that a periodic wave consists of multiple frequency components, that transient signal may have high frequency components, and that ground motion, whose peak values are often utilized in earthquake engineering, usually has higher-frequency components. A sampling frequency of 256 Hz is a reasonable assumption for SHM system with smart sensors. When a 0.1 Hz component of a signal sampled at 256 Hz is analyzed with an FFT of length 1,024, only less than a half of the natural period fits in one window; the resolution of FFT is too coarse. In such cases, the sampling frequency is lowered or the number of FFT points is increased. Increasing the number of FFT points results in an increase in the total number of data points.

The data type also affects the amount of memory required to store a measured signal. The most often utilized are 16- and 32-bit integers, and double precision data. The sizes of these data types are 2, 4, and 8 bytes, respectively. The effective number of bits of resolution in the measured data determines the requirements on the data type. For example, an acceleration signal with a resolution of 20 effective bits needs to use either a 32-bit integer or double precision data. Considering that many ADCs on smart sensors has resolution poorer than 16-bit, 16-bit integers are most likely able to represent measured sensor data.

The combination of 21,504 data points and 16-bit integer yields 43 kB of data per sensing channel. Though this number will vary with changes in the number of data points or the data type, 43 kB is used from here on as a typical amount of data generated per sensing channel.

The number of sensing channels also depends on the applications. For example, one of the bridges in Hong Kong famous for its densely instrumented sensors, Ting Kau Bridge, has 67 channels to measure acceleration and 132 channels to measure strain. These channels are normally sampled at 25.6 Hz. As such, hundreds of sensor nodes are considered herein. One sensor node may have multiple sensing channels; for example, acceleration measurement in three directions needs three channels. In total, this research has a target of approximately a thousand channels.

A thousand sensing channels, each of which generates 43 kB of data, produce 43 MB of data per measurement. Centrally collecting such a large amount of data is not practical. Mechitov et al. (2004) reported that communication at the sink node is the bottle neck and that collection of 480 kB of data from 16 Mica2s required more than 6,000 seconds. If smart sensors with limited hardware resources, especially battery and RF components, are used for SHM applications, transfer and processing of this large amount of data need to be well-planned.

5.1.2 Model-based data aggregation

The amount of data involved in SHM applications normally exceeds practical communication capabilities of smart sensor networks if all of the measurement data needs to be collected centrally; data aggregation has been an important issue to be addressed before an SHM system employing smart sensors is realized. One approach to overcome this data aggregation problem has been an independent data processing strategy (i.e., without internode communication). This approach, however, cannot fully exploit information available in the sensor network. Distribution of data processing and coordination among smart sensors play a central role in addressing many smart sensor implementation issues, including data aggregation. This section will demonstrate that distribution and coordination can be well-planned so that the data aggregation problem is addressed without sacrificing performance of the SHM algorithms.

The auto-correlation and cross-correlation functions are the inverse Fourier transform of the PSD and CSD functions, respectively; their estimation is the beginning step of many output-only SHM algorithms. CSD estimation requires data from two sensor nodes. Measured data needs to be transmitted from one node to the other before data processing takes place. Associated data communication can be prohibitively large without careful consideration of the implementation. Distributed estimation of correlation functions is proposed in this section.

Correlation functions are, in practice, estimated from finite length records. PSD and CSD functions are estimated first through the following relation (Bendat & Piersol, 2000):

$$\hat{G}_{xy}(\omega) = \frac{1}{n_d T}\sum_{i=1}^{n_d} X_i^*(\omega) Y_i(\omega) \tag{5.1}$$

where $\hat{G}_{xy}(\omega)$ is an estimate of CSD $G_{xy}(\omega)$ between two stationary Gaussian random processes, $x(t)$ and $y(t)$. $X(\omega)$ and $Y(\omega)$ are the Fourier transforms of $x(t)$ and $y(t)$; * denotes the complex conjugate. T is time length of sample records, $x_i(t)$ and $y_i(t)$. When $n_d = 1$, the estimate has large random error. The random error is reduced by computing an ensemble average from n_d different or partially overlapped records. The normalized RMS error $\varepsilon(|G_{xy}(\omega)|)$ of the spectral density function estimation is given as

$$\varepsilon(|\hat{G}_{xy}(\omega)|) = \frac{1}{|\gamma_{xy}|\sqrt{n_d}} \tag{5.2}$$

$$\gamma^2{}_{xy}(\omega) = \frac{|G_{xy}(\omega)|^2}{G_{xx}(\omega) G_{yy}(\omega)} \tag{5.3}$$

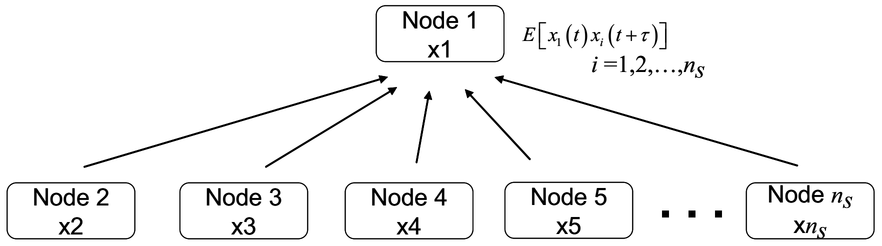


Figure 5.1. Centralized correlation function estimation.

$\gamma^2_{xy}(\omega)$ is the coherence function between $x(t)$ and $y(t)$, indicating the linear dependence between the two signals. Through the averaging process, the estimation error is reduced. Averaging 10 to 20 times is common practice. The estimated spectral densities are then converted to correlation functions via the inverse Fourier transform.

An implementation of correlation function estimation for a small community of sensors in a centralized data collection scheme is shown in Figure 5.1, where node 1 works as a reference sensor. Assuming n_s nodes, including the reference node, are measuring structural responses, each node acquires data and sends it to the reference node. The reference node calculates the spectral density as in Eq. (5.1). This procedure is repeated n_d times and averaged. After averaging, the inverse Fourier transform is taken to calculate the correlation function. All calculations take place at the reference node. When the spectral densities are estimated from discrete time history records of length N, the total data to be transmitted over the network using this approach is $N \times n_d \times (n_s - 1)$.

In the next scheme, data flow for correlation function estimation is examined and data transfer is reorganized to take advantage of computational capability on each smart sensor node. After the first measurement, the reference node broadcasts the time record to all of the nodes. On receiving the record, each node calculates the spectral density between its own data and the received record. This spectral density estimate is locally stored. The nodes repeat this procedure n_d times. After each measurement, the stored value is updated by taking a weighted average between the stored value and the current estimate. In this way, Eq. (5.1) is calculated on each node. Finally the inverse Fourier transform is taken of the spectral density estimate locally. The resultant correlation function is sent back to the reference node. Because subsequent modal analysis such as ERA uses, at most, half of the correlation function data length, $N/2$ data points are sent back to the reference node from each node. The total data to be transmitted in this scheme is, therefore, $N \times n_d + N/2 \times (n_s - 1)$ (see Figure 5.2).

As the number of nodes increases, the advantage of the second scheme, in terms of communication requirements, becomes significant. The second approach requires data transfer of $\mathbf{O}(N \cdot (n_d + n_s))$, while the first one needs to transmit to the reference sensor node data of the size of $\mathbf{O}(N \cdot n_d \cdot n_s)$. The distributed implementation leverages knowledge regarding the application to reduce communication requirements as well as to utilize the CPU and memory in a smart sensor network efficiently.

The data communication analysis above assumes that all the nodes are in single-hop range of the reference node. This assumption is not necessarily the case for a general SHM

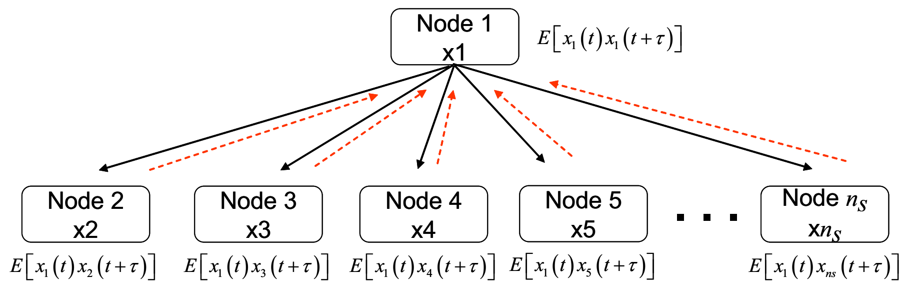


Figure 5.2. Distributed correlation function estimation.

application. However, Gao (2005) proposed a DCS for SHM which supports this idea. Neighboring smart sensors in a single-hop communication range make local sensor communities and perform SHM in the communities. In such applications, the assumption of nodes being within single-hop range of a reference node is reasonable.

5.1.3 Implementation

In this section, the data aggregation strategy introduced previously is validated on a network of Imote2s. To understand the performance of the algorithms, consider the truss model shown in Figure 5.3. Vertical acceleration responses under random input are measured at nodes 9, 11, 13, 15, 17, 19, 21, 23, and 25. The responses are then injected into a network of nine Imote2s. The Imote2 corresponding to node 9 works as the reference sensor. This node broadcasts its own data to the other nodes and collects correlation function estimates. The length, N, of the injected data is 2,048 and the number of FFT points is 512, resulting in seven averages with a 50 percent overlap. The sampling rate is 500 Hz.

The acceleration response data and correlation function estimates are stored and transferred as a 16-bit integer data type, considering that accelerometer can reasonably be assumed to have a 12-bit resolution. In this way, memory space and bandwidth are efficiently utilized, while double precision data is used during the spectral density estimation and the inverse FFT calculations.

Simulation data is then reliably transferred to the destination nodes using a reliable communication protocol described in the next section. In this way, outputs of this correlation function estimate are compared with those calculated on a PC without being mixed with the effect of data loss.

Correlation functions are estimated on the Imote2s and reported back to the reference node. Figure 5.4 shows the estimated correlation function between nodes 9 and 11. This estimate matches the corresponding correlation function estimated on a PC assuming the

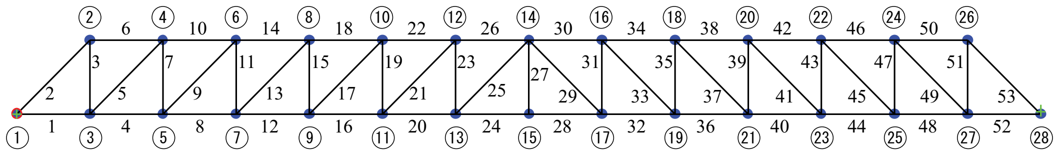


Figure 5.3. Numerical model of a 2D truss structure.

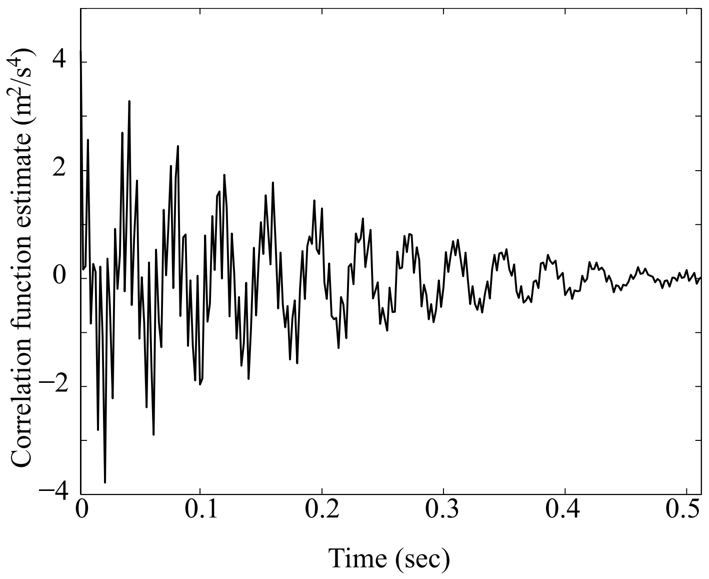


Figure 5.4. A correlation function estimate.

associated quantization error. Because the difference between these two signals is smaller than 10^{-13}, only the estimate on the Imote2 is shown on Figure 5.4.

Data transferred in a network is reduced as explained in section 5.1.2. In this example, data transmission is reduced by a factor of 5, as compared to centralized correlation function estimation implementation. This reduction factor can be larger depending on the number of sensors and the associated averaging. This reduction shows the advantage of the distributed correlation function estimation.

Further consideration is necessary to accurately assess the efficacy of the distributed implementation. Power consumption of smart sensor networks is not simply proportional to the amount of data transmitted. Acknowledgment messages are also involved. The radio listening mode consumes power, even when no data is received. However, the size of the measured data is usually much larger than the size of the other messages to be sent and should be considered the primary factor in determining power consumption. Small data transfer requirements realized by the proposed model-based data aggregation algorithm will lead to reduced power consumption.

5.2 Reliable communication

RF communication is not reliable unless lost packets are specifically addressed. Packets may not be transmitted properly. When the distance between nodes is too long, packets may not reach the destination. Multiple nodes trying to send packets at the same time can cause packet collisions. If packets carrying commands are lost, destination nodes fail to perform certain tasks. The sender is unsure whether the destination nodes have received commands. If packets carrying measurement data are lost, destination nodes cannot fully reconstruct the sender's data. Therefore, packet loss may cause a system to be in an unknown state and may degrade measurement signals. SHM applications employing smart sensors must address packet loss.

When smart sensor applications involve more and more complicated internode data processing and are assigned more and more tasks by commands sent through packets, these commands needs to be delivered reliably. Reliable communication of short messages is clearly a significant help in developing SHM systems with complicated internode data processing.

The need to transfer a large amount of data reliably is not necessarily clear. In many SHM research efforts, the data loss problem is not addressed. Loss of a few data points has often been considered acceptable. However, the rationale behind accepting a small packet loss rate is not clear. In section 5.2.1, the effects of data loss on SHM applications are assessed.

The packet loss rate of the Imote2 is then experimentally examined. The packet loss rate varies from experiment to experiment. An experiment with nodes close to each other is expected to have less packet loss than an experiment with sparsely distributed nodes. Packet loss rate is estimated under several conditions in section 5.2.2

Subsequently, reliable communication protocols suitable for sending large amounts of data, as well as protocols to send single packets, are proposed.

5.2.1 The effects of data loss on SHM applications

In a wireless network, some packets are inevitably lost during communication unless the communication protocol is specifically designed for reliable communication. Conventional statistical, modal, and structural analyses of structural response data, however, assume that no loss of data takes place. Some researchers have been working to develop reliable communication without data loss, while others just ignore the data loss effects on their analyses. Modal analysis and damage localization has not yet been examined from the perspective of data loss. The impact of data loss on SHM applications is investigated herein.

The Distributed Computing Strategy (DCS) for SHM described in section 6.4 is used as a benchmark application. The correlation function and impulse response function estimation, as well as modal analysis, employed as a part of DCS are widely used to analyze ambient vibration data of civil infrastructure. The outcome of this data loss analysis is applicable to many vibration-based SHM applications. The damage detection method adapted in DCS is the DLV method, and the effect of data loss on this damage detection method is also investigated. Understanding the effect of data loss may provide insight into how to accommodate communication with data loss that is less demanding on resource limited smart sensors than the communication without data loss (Nagayama et al., 2007).

A computer simulation study is conducted for a truss model (see Figure 5.3), assuming various data loss levels to investigate the data loss effect. Smart sensors are assumed to be placed at the 13 nodes on the lower chord to measure the vertical acceleration. The vertical input excitation at node 11 is measured. The sampling frequency is set at 380 Hz so that the Nyquist frequency is above the fourth natural frequency of the structure. After the data is acquired, a certain percentage of the data is randomly dropped

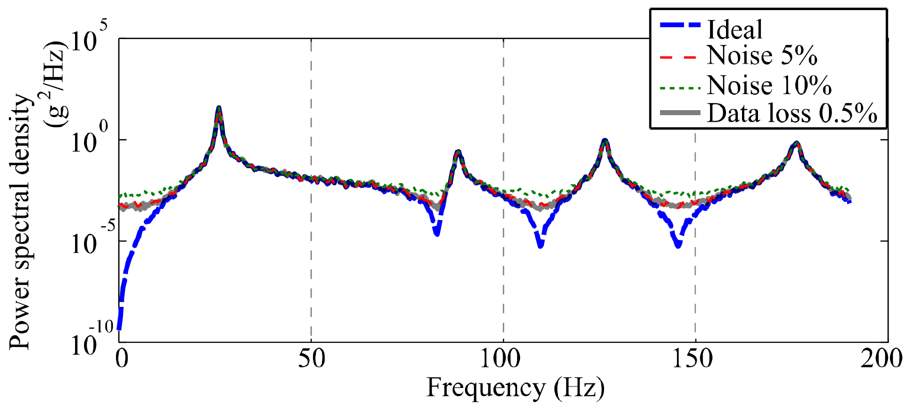


Figure 5.5. Effect of data loss and observation noise: acceleration power spectral density.

to simulate data loss. Data loss is assumed to take place only at this step, where the largest amount of data is transferred among the sensor nodes.

Data is processed following each step of DCS. The impulse response functions associated with each measurement points are estimated. Then the impulse response functions are fed into the ERA routine for modal analysis. The natural frequency, damping ratio, and mode shapes are among the major outputs. These modal parameters yield flexibility matrix estimations. Then, one of the truss elements of the structure, element 9, is replaced with a damaged element, which has a 40 percent cross-section reduction. Data acquisition, impulse response estimation, modal analysis, and flexibility matrix estimation are repeated on this data. From the estimated flexibility matrices for an undamaged and damaged model, the DLVs are calculated. The accumulated stress, small values of which indicate possible damaged elements, is then calculated. This simulation is conducted assuming several data loss levels.

For the results of the data loss analysis to be better interpreted, the outcome is compared to that of computer simulation including observation noise (but without data loss). A band-limited white noise is superimposed onto each of observed signals. RMS noise level is specified as a percentage of the RMS of physical responses.

The effect of data loss is first examined by comparing the PSD and coherence functions of the measurement data. Figure 5.5 shows representative PSD functions calculated with and without noise/data loss. The PSD's peaks remain nearly unchanged when the data loss or the noise is introduced; however, zeros are blurred. Although further investigation is necessary for quantitative judgment, these results indicate that data loss of 0.5 percent and 5 percent observation noise have similar impact on PSD estimation.

Coherence functions indicate the degree of linearity between two variables. In this computer simulation, the input excitation is applied only at node 11. Because the response of the truss structure is linear, the coherence function between the two measurement signals is expected to be unity over the entire frequency range. When no data loss and no observation noise are considered, the coherence function is indeed close to one, as shown

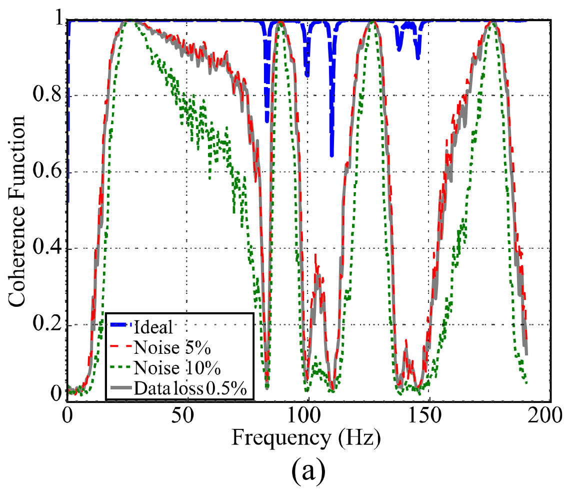
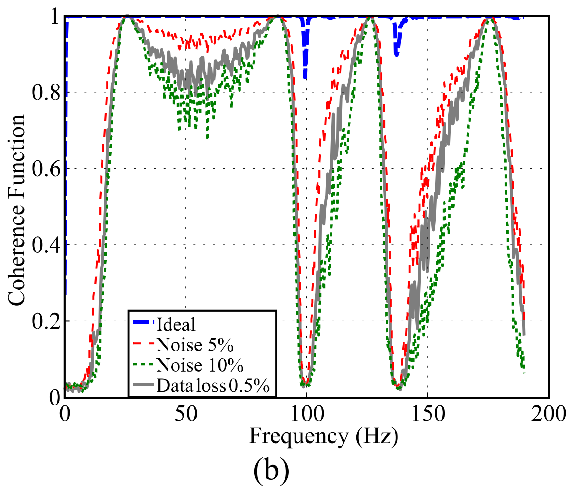


Figure 5.6. Effect of data loss and observation noise: coherence functions between (a) nodes 11 and 19; and (b) nodes 19 and 25.



in Figure 5.6. The isolated downswings correspond to zeroes of the system; extremely small responses of the structure at these frequencies at the zeros result in numerical error, giving small coherence function.

The loss of 0.5 percent of data yields a much smaller coherence function, except at the system's poles (see Figure 5.6). Because of the random nature of data loss occurrence and the excitation, the coherence function varies from simulation to simulation. Twenty time average of the coherence functions are shown in this figure. The coherence function's discrepancy from unity depends on the two measurement nodes between which the function is calculated. All of the investigated coherence function plots, however, support that the loss of 0.5 percent of data affects the coherence function in a similar way as 5 to 10 percent measurement noise addition.

Though the data loss clearly impacts the PSD estimation and coherence function negatively, the consequence in subsequent modal analyses is unclear. The modal properties are mainly represented by frequency components near the natural frequency. The PSD and coherence function are not affected by data loss and noise as much near the poles. The ERA modal analysis method is then applied to the impulse response functions to estimate the modal properties. Impulse response functions are the inverse Fourier transform of the transfer functions from the input force to measured outputs, which is calculated from CSD and PSDs.

Figure 5.7 shows how the estimates of the first natural frequency and the damping ratio vary as a function of the data loss level. For comparison, these modal parameters are also estimated from the simulation model subjected to measurement noise (see Figure 5.8). Again, a loss of 0.5 percent of data is approximately equivalent to 5 to 10 percent measurement noise.

The first four modes are identified and these modal properties are used to estimate the flexibility matrix $\mathbf{F}_0$. The flexibility matrix estimate $\hat{\mathbf{F}}$ is compared with $\mathbf{F}_0$ and the estimation error is shown in Figure 5.9 in terms of the Frobenius norm. $\| \, \|_F$ represents the

Frobenius norm. Because only a limited number of modes are utilized in the flexibility matrix estimation, the Frobenius norms do not approach zero at 0 percent data loss and 0

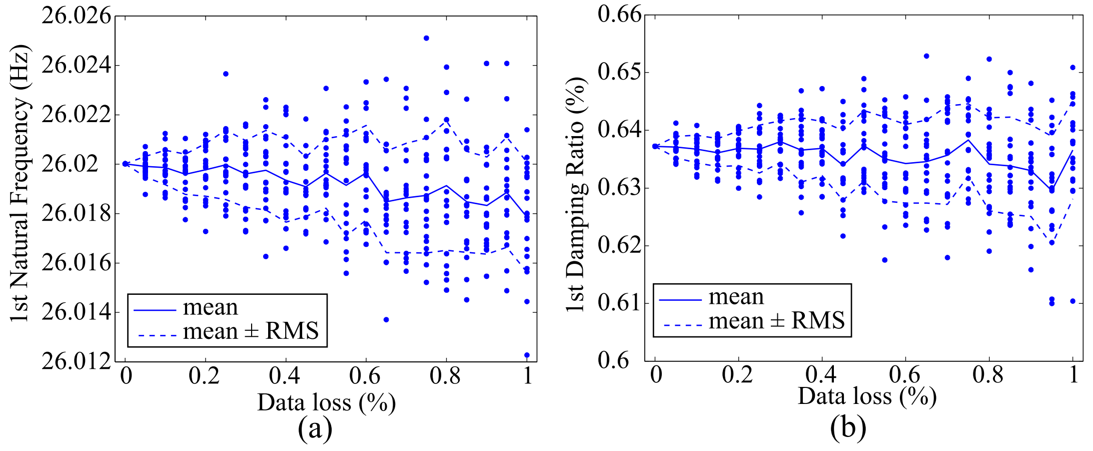


Figure 5.7. Data loss dependency of (a) the 1st natural frequency; and (b) damping ratio.

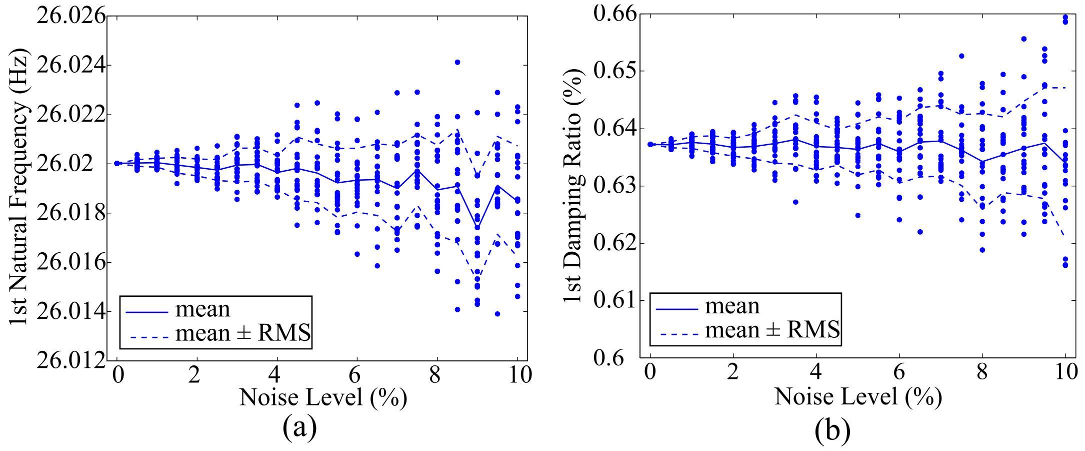


Figure 5.8. Noise level dependency of (a) the1st natural frequency; and (b) damping ratio

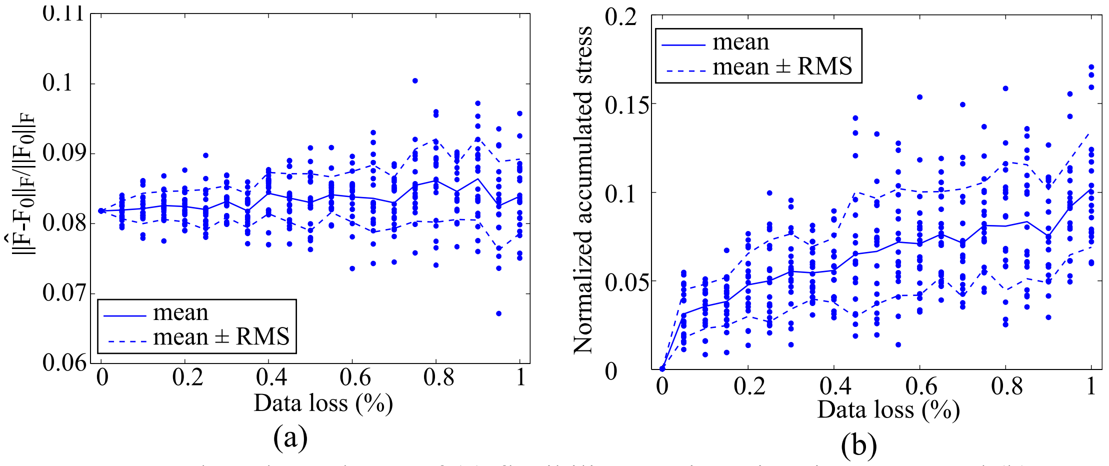


Figure 5.9. Data loss dependency of (a) flexibility matrix estimation error; and (b) normalized accumulated stress in the damaged element.

percent noise. The flexibility matrix is estimated both before and after damage is introduced, and then the DLV method is applied to these matrices. The accumulated stress calculated in the DLV method is an indication of damage. The stress in a damaged member should be zero. The accumulated stress in the damaged element is plotted on Figure 5.9, together with the flexibility matrix estimation error. The stress is indeed zero for 0 percent data loss. As the amount of data loss increases, the accumulated stress becomes larger. A stress value around 0.1 is considered small, indicating that the element is damaged. The estimation error for these quantities due to noise is shown in Figure 5.10 for comparison. From these figures, a loss of 0.5 percent data is considered to be equivalent to 5 to 10 percent of measurement noise.

From this analysis, data loss is found to introduce error into the modal analysis; the error affects the subsequent analysis. As data loss increase, the quality of data degrades. Because of the statistical approach (i.e., averaging) associated with the PSD and the CSD estimation, a certain amount of data loss can be accommodated. A loss of 0.5 percent data might be acceptable, considering that a corresponding 5 to 10 percent observation noise is unexceptional in the monitoring of civil infrastructure.

5.2.2 Packet loss estimation in RF communication

The packet loss rate is experimentally estimated in this section. Together with the results from the previous section, experimental data are examined to see whether the packet loss rate is acceptable for SHM applications.

1. Experimental setup

Eight Imote2s, one programming board, and a PC were configured to perform this experiment. One of the Imote2s is programmed as the base station. Another Imote2 works as the sender. The other six Imote2s are configured to be receivers. A java program running on the PC reads parameters from an input file and sends commands to the base station node. The base station receives information about the number of receivers, node

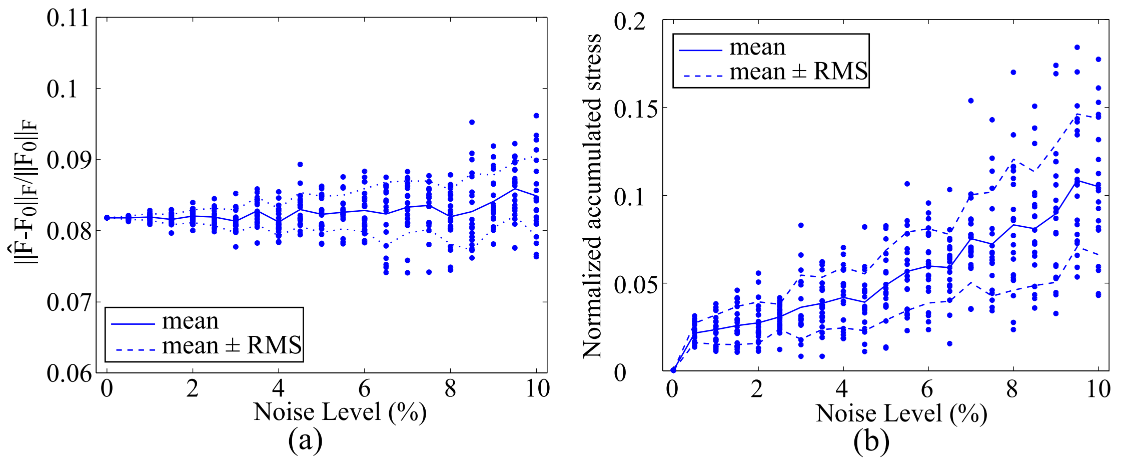


Figure 5.10. Noise level dependency of (a) flexibility matrix estimation error; and (b) normalized accumulated stress in the damaged element.

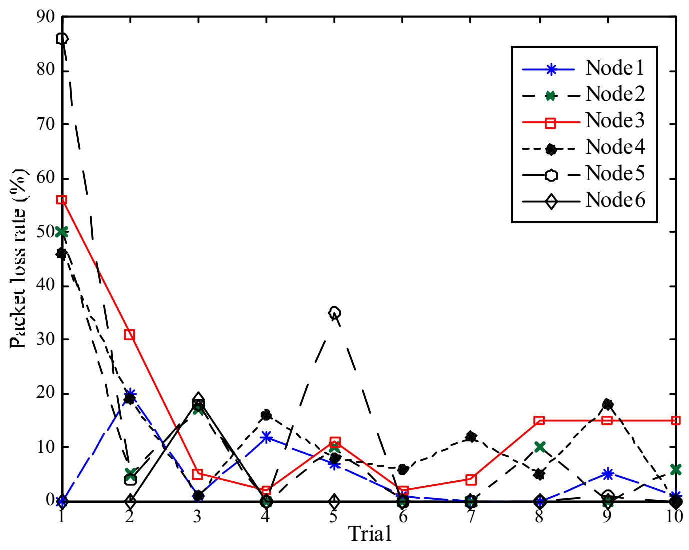


Figure 5.11. Packet loss rate estimation.

IDs of the sender and receivers, the number of packets to be sent, and the number of repetitions. The base station forwards this information to the sender. On reception, the sender starts broadcasting packets. In this experiment, 100 packets are broadcast. After the sender completes transmission of the 100 packets, the sender tells the receiver that transmission is complete and queries each receiver regarding how many packets were received. This procedure is repeated 10 times.

This experiment is conducted on the lawn in front of Newmark Civil Engineering Laboratory at the University of Illinois at Urbana-Champaign. The sender and receiver nodes are held at the height of about 1 m. The distance between the sender and the receiver is 3.3 m. Figure 5.11 summarizes the results. In this experiment, the packet loss is usually smaller than 20 percent. There are many rounds of data transfer without any lost packets, while the maximum packet loss rate reached 86 percent. The packet loss rate does not show a clear trend among the six nodes. One node without any packet loss in one round of data transfer may suffer from severe data loss in the subsequent round of communication. While a loss of no or a few packets is anticipated to take place often in communication, such a low packet loss rate cannot always be assumed. Smart sensor communication sometimes experiences a loss of a large number of packets.

A packet loss of 20 percent or 86 percent is apparently much larger than the data loss level discussed in the previous section and is not acceptable for SHM applications. A reliable communication protocol suitable for transfer of a large amount of data, as well as a protocol for short messages, is proposed in the next section.

5.2.3 Reliable communication protocol

Reliable communication protocols for long data records and short messages are developed in this section. Communication packets range from one-bit acknowledgments to lengthy acceleration time histories. A communication protocol suitable for long data records is not necessarily efficient for single packet transfer. Communication protocol for long data records and single packets are, therefore, designed separately.

In the SHM architecture employed in this research, most communication takes place inside a smart sensor cluster as unicast or multicast. Unicast is communication between one sender and one receiver, while multicast is communication from one sender to multiple receivers. Both unicast and multicast protocols are, therefore, developed for long data records and for short message communication. Communication protocols developed in this chapter suffice for most communication requirements in DCS for SHM with the exception being communication involved in time synchronization.

1. Communication protocol for long data records

Communication protocols suitable to transfer long data records reliably are developed. The length of data assumed in this section is the acceleration record length used in DCS. The first step of DCS estimates the CSD using FFT. The estimation is frequently based on 10 to 20 times the number of data points used in the FFT. Assuming the number of FFT points is 1,024, the target length of data to be transmitted is set as 11,264.

Though repeated data transmission without acknowledgment can statistically improve the reliability of communication, such protocols cannot guarantee communication success rate deterministically. If the packet loss rate is expected to be approximately constant over long time, repeated sending practically results in no data loss; the number of repetitions can be dynamically adjusted based on the packet loss rate of the last communication. When many packets are lost statistically, the number of repetitions can be increased. However, in smart sensor networks, burst packet loss may take place. RF transmission devices nearby, for example, may cause the large number of communication packets to be dropped. The fluctuation in packet loss rate is considered large, as shown in Figure 5.11. Repeated transmission is not a sufficient solution to the packet loss problem.

Acknowledgment packets potentially guarantee reliable communication between nodes. However, a poorly-designed communication protocol involving acknowledgment messages can be notably inefficient. Many acknowledgment messages may be required, waiting times may be long, and the same packets may need to be sent many times. With careful consideration of efficiency, reliable communication protocols are realized in this section using acknowledgment messages. These protocols are similar to error control methods for data transmission, which are briefly reviewed and then modified for use in the proposed reliable communication protocol for SHM applications.

Automatic Repeat reQuest (ARQ) is an error control method which repeats sending packets based on the request from the receiver. On reception of packets without error, the receiver replies with a positive acknowledgment (ACK). If an error is detected, the receiver sends a negative acknowledgment (NACK) and requests retransmission. There are several ARQ protocols, some of which are described in the following paragraphs.

In the stop-and-wait protocol, the sender waits for an acknowledgment after the transmission of each packet. If an ACK is received, the sender transmits the next packet.

On reception of a NACK, the sender resends the last packet. Simplicity is a main advantage of this method. The IEEE 802.11 wireless LAN adapts the stop-and-wait protocol. This protocol is, however, not efficient in the sense that the sender needs to wait for acknowledgment of each packet.

In the go-back-N protocol, the sender transmits more than one packet without waiting for acknowledgment. The receiver keeps receiving packets replying ACK until an error is detected. If an error is found in the N-th packet, the N-th packet and all subsequent packets are discarded. The receiver sends back a NACK with the number corresponding to the lost packet. The sender resends packets from the N-th packet. Discarding the received packets after the N-th packets lowers the efficiency of this protocol.

Selective-repeat protocol addresses the inefficiency in the go-back-N protocol. The sender continues sending unsent packets. Once the receiver detects an error in a packet, a NACK is sent back to the sender. Packets received after detection of an error are stored in a reception buffer. Only the packet with an error is disregarded. On reception of a NACK, the sender temporary stops sending unsent packets and resends the lost packet. Then the sender continues to send the remaining packets.

These concepts of ARQ can be utilized to guarantee reliable communication among smart sensors. The reliable communication protocol developed by Mechitov et al. (2004) sends a set of packets and then waits for acknowledgment. If the receiver does not receive ACK, the same set of packets is sent again. Upon reception of ACK, the sender moves on to the next set of packets. Even when one of packets sent in a group is lost, all of the packets are sent again, reducing the efficiency of the protocol.

In this section, concepts from the ARQ protocol are modified for reliable wireless communication. The RF component on the Imote2 is in either a listening mode or a transmission mode. During transmission, the Imote2 cannot receive packets. However, in selective-repeat protocol, transmission and reception are deeply interwoven. The receiver may send an acknowledgment while the sender is in transmission mode. Packet loss and retransmission are expected to be more frequent if the ARQ protocol is implemented directly. Scheduling interwoven transmission and reception may result in long waiting times. In the proposed protocols, the sender transmits all of the packets before this node expects an acknowledgment packet. The receiver stores all of the received data in a buffer. Once the receiver gets a message indicating the end of data transmission, the receiver replies to the sender, indicating which packets are missing. Only missing packets are sent again. In this way, the number of acknowledgments and retransmissions can be greatly reduced.

This protocol is designed to send either 64-bit double precision data, 32-bit integers, or 16-bit integers. Many ADCs on traditional data acquisition systems have a resolution less than 16 bits, supporting the need for transfer of 16-bit integer format data. Some ADCs have a resolution better than 16 bits, necessitating data transfer in 32-bit integer format. Once an acceleration record is processed, the outcome may need more bits. Onboard data processing such as FFTs and SVDs is usually performed using double precision calculations. Even when the effective number of bits is smaller than 32, debugging of onboard data processing greatly benefits from transfer of double precision

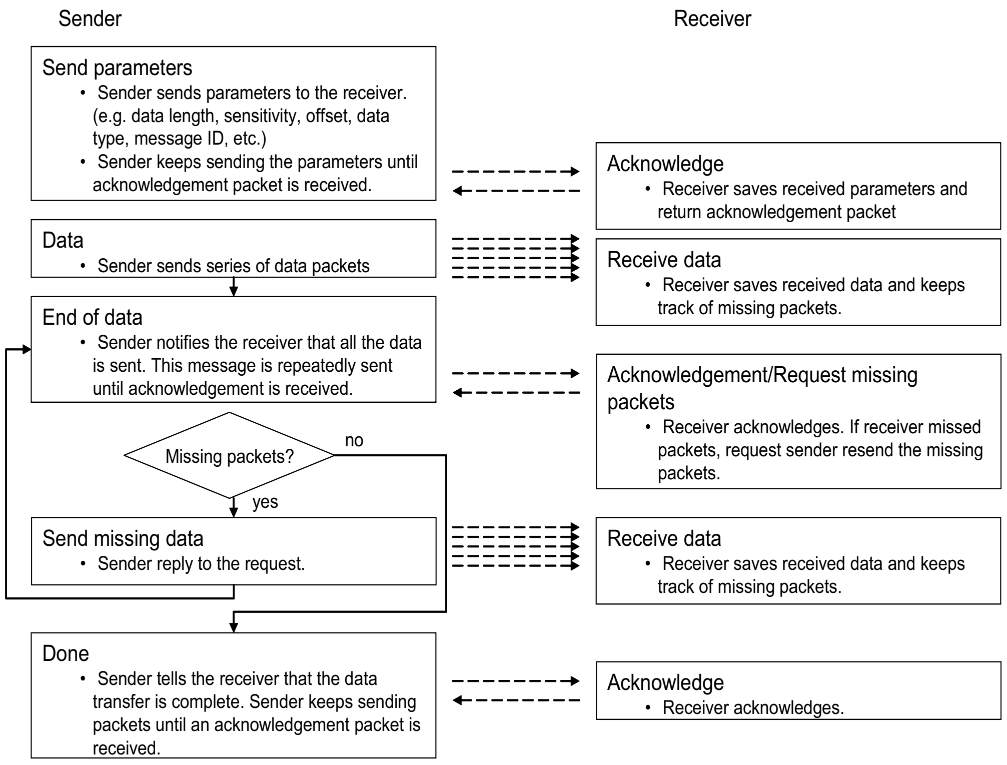

Sender
Receiver
Send parameters
• Sender sends parameters to the receiver. (e.g. data length, sensitivity, offset, data type, message ID, etc.)
• Sender keeps sending the parameters until acknowledgement packet is received.
Acknowledge
• Receiver saves received parameters and return acknowledgement packet
Data
• Sender sends series of data packets
Receive data
• Receiver saves received data and keeps track of missing packets.
End of data
• Sender notifies the receiver that all the data is sent. This message is repeatedly sent until acknowledgement is received.
Acknowledgement/Request missing packets
• Receiver acknowledges. If receiver missed packets, request sender resend the missing packets.
Missing packets?
no
yes
Send missing data
• Sender reply to the request.
Receive data
• Receiver saves received data and keeps track of missing packets.
Done
• Sender tells the receiver that the data transfer is complete. Sender keeps sending packets until an acknowledgement packet is received.
Acknowledge
• Receiver acknowledges.

Figure 5.12. Simple block diagram of the reliable unicast protocol for long data records.

data; data processing results on Imote2s can be directly compared with those on a PC, which are most likely in double precision format. Transfer of 64-bit double precision data is supported based on such needs.

Unicast

In the unicast protocol, the sender seeks to reliably deliver long data records to the receiver utilizing acknowledgment messages. Figure 5.12 shows a simple flowchart of this protocol.The details of the protocol are explained in the following paragraphs as well as in the block diagram in Figure 5.13.

On start-up, the communication protocol requests that the main application assign a buffer space to keep data to be transferred. This buffer needs to be large enough to hold the entire data record. When 11,264 data points in 16-bit integer format are transferred, this buffer size is about 22.5 kB. The main application returns the pointer to this buffer. The need for this large buffer size is a drawback of this approach. However, with knowledge of the application, the main application can utilize this memory space for other purposes when communication is not expected to take place. This buffer is allocated in the main application and only the pointer is given to the communication program.

When the application is ready to send data, the sender informs the receiver of the necessary parameters (e.g., data length, data type, message ID, destination node ID, source node ID, etc.). If the data is in integer format, the sensitivity and offset of the data are also conveyed to the receiver. A packet containing these parameters is sent to the receiver.

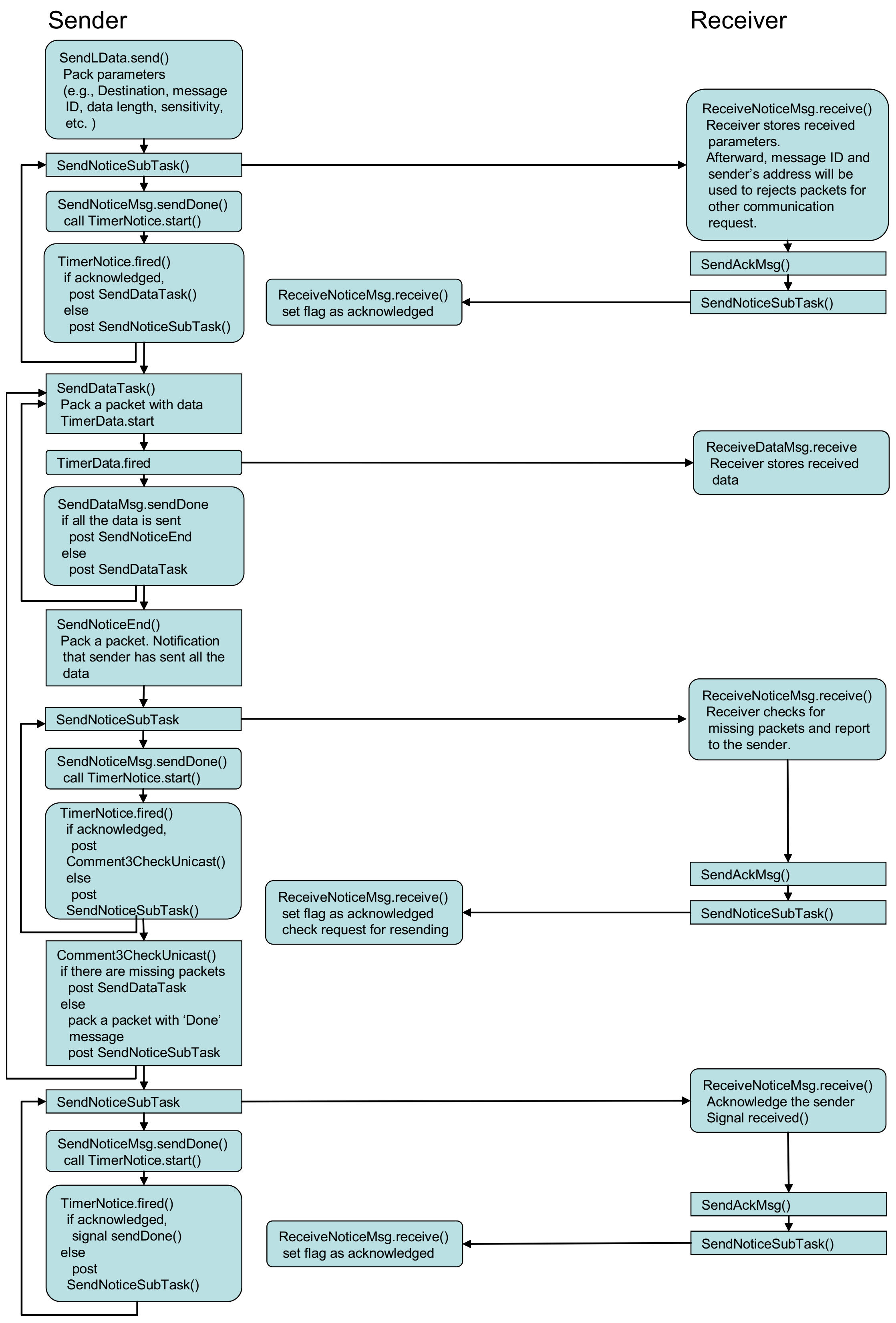

Sender
Receiver
SendLData.send() Pack parameters (e.g., Destination, message ID, data length, sensitivity, etc.)
SendNoticeSubTask()
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, post SendDataTask() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() Receiver stores received parameters. Afterward, message ID and sender's address will be used to rejects packets for other communication request.
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged
SendDataTask() Pack a packet with data TimerData.start
TimerData.fired
ReceiveDataMsg.receive Receiver stores received data
SendDataMsg.sendDone if all the data is sent post SendNoticeEnd else post SendDataTask
SendNoticeEnd() Pack a packet. Notification that sender has sent all the data
SendNoticeSubTask
ReceiveNoticeMsg.receive() Receiver checks for missing packets and report to the sender.
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, post Comment3CheckUnicast() else post SendNoticeSubTask()
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged check request for resending
Comment3CheckUnicast() if there are missing packets post SendDataTask else pack a packet with 'Done' message post SendNoticeSubTask
SendNoticeSubTask
ReceiveNoticeMsg.receive() Acknowledge the sender Signal received()
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, signal sendDone() else post SendNoticeSubTask()
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged

Figure 5.13. Detailed block diagram of the reliable unicast protocol for long data records.

After this transmission, the sender periodically checks whether an acknowledgment from the receiver has arrived. If not, this packet is sent again. The message ID of the sender is incremented when the sender engages in the new round of reliable communication. Once two nodes engage in a round of reliable communication, packets with different message IDs, destination addresses, or source addresses are disregarded.

To appropriately interpret the received packet, the Imote2 uses a 1-byte comment on each packet. For example, a packet from the sender with the comment '1' is interpreted as the first packet with parameters such as data length, data type, etc. On reception of a packet, tasks predetermined for the comment are executed.

Also, the current state of the sender and the receivers are internally maintained using another 1-byte variable on each node. For example, the sender's current state is '1' after sending the packet with the comment '1'. This state changes to '2' when the sender finds that an acknowledgment for this packet is received. The tasks to be executed in events such as packet reception and timer firing are determined based on the current state and the comment on the most recently received packet.

Upon reception of a packet with the comment '1' from the sender, the receiver checks whether it is ready to receive data. Unless this node has already agreed to receive data from another node or send data to another node, this node replies to the sender with an acknowledgment that it will receive the data from the sender.

When the sender notices that the acknowledgment has arrived, the periodical check for acknowledgment stops and the entire data record is sent to the receiver. One packet contains either three double precision data points, six 32-bit integers, or twelve 16-bit integers. Each packet also contains a 2-byte node ID for the sender and a 2-byte packet number, which the receiver uses to reconstruct the original data in the buffer. To keep track of received packets, the receiver has an array, which is initialized to zeroes on reception of a packet with the comment '1'. One bit of the array is assigned to each data packet. Upon reception of each packet, the corresponding bit is changed to one. By examining this array and the total number of packets calculated from the data type and the total length of data, the receiver knows which packets are missing.

When the sender finishes transmitting all of the data, the end of the data is signaled to the receiver by sending a packet with the comment '2'. The sender periodically checks the response from the receiver. If nothing is received, this packet is sent again.

Knowing that all of the data were transmitted from the sender, the receiver checks the array to keep track of packet reception. If all of the packets are received, an acknowledgment is returned to the sender. The sender tells the receiver that the data transfer is complete by returning a packet with the comment '4' to the receiver. This message is also resent until acknowledgment is returned from the receiver. When the receiver first acknowledges the packet with the comment '4', the receiver signals to the main application on the receiver that a set of data has been successfully received, thus completing the receiving activity on this node. Once the sender receives this acknowledgment, the main application on the sender is notified that the data has been sent successfully, thus completing the sending activity on this node.

If there are missing packets, the receiver responds to the end-of-data notice by sending a packet with the missing packets' IDs. The first eight missing packets' IDs are sent to the sender. The sender packs these missing packets and sends them to the receiver. At the end of transmission, the packet with the comment '2' is sent again. If there are missing packets, the same procedure is repeated. If all of the data is received, an acknowledgment is returned, and the communication activity is completed as described in the previous paragraph.

One of the difficulties with this protocol is to judge when communication ends. The sender can clearly know the end of communication when the acknowledgment message with the comment '4' is received. However, the receiver cannot tell the end of communication in a strict sense. The receiver cannot know whether the acknowledgment with the comment '4' reached the sender or not. If the receiver does not receive any packet after the transmission of the acknowledgment packet with the comment '4', there are two possibilities: The acknowledgment packet reached the sender and this round of communication was successfully completed; or the acknowledgment packet did not reach the sender and the sender is periodically transmitting packets with comment '4', none of which reaches the receiver. There is no way to judge between the two possibilities.

This inability to judge the end of communication may cause a serious problem. If only the receiver is disengaged from the communication, then the sender will keep sending the last packet to the receiver; the receiver never replies because the receiver is already disengaged.

This problem is addressed by storing message and source IDs of the last 15 rounds of reliable communication and separating the last acknowledgment process from the rest of the process. When the receiver gets the packet with the comment '4' for the first time, this node sends an acknowledgment back and is disengaged from this communication activity. The message and source IDs are stored on this node. If this node later receives the same packet, the structure storing the message and source IDs is searched. Once the receiver finds the message and source IDs of the received packet in the structure, an acknowledgment is sent back to the sender. This acknowledgment with the comment '4' which is returned to the sender is sent using a message type assigned only for this purpose. Even when the receiver is engaged in the next round of communication, this process of acknowledgment can be done without affecting the subsequent or ongoing communication activities or internal variables.

Multicast

In the multicast communication protocol, the sender reliably transmits long data records to multiple receivers utilizing acknowledgment messages. Figure 5.14 shows a simple flowchart of this protocol. The details of this protocol are explained in the following paragraphs as well as in the block diagram in Figure 5.15.

The primary difference between the multicast and unicast protocols is the first step. Before the sender sends the required parameters, such as data length and data type, the node IDs of the receivers are broadcast, as well as the source and message IDs. The

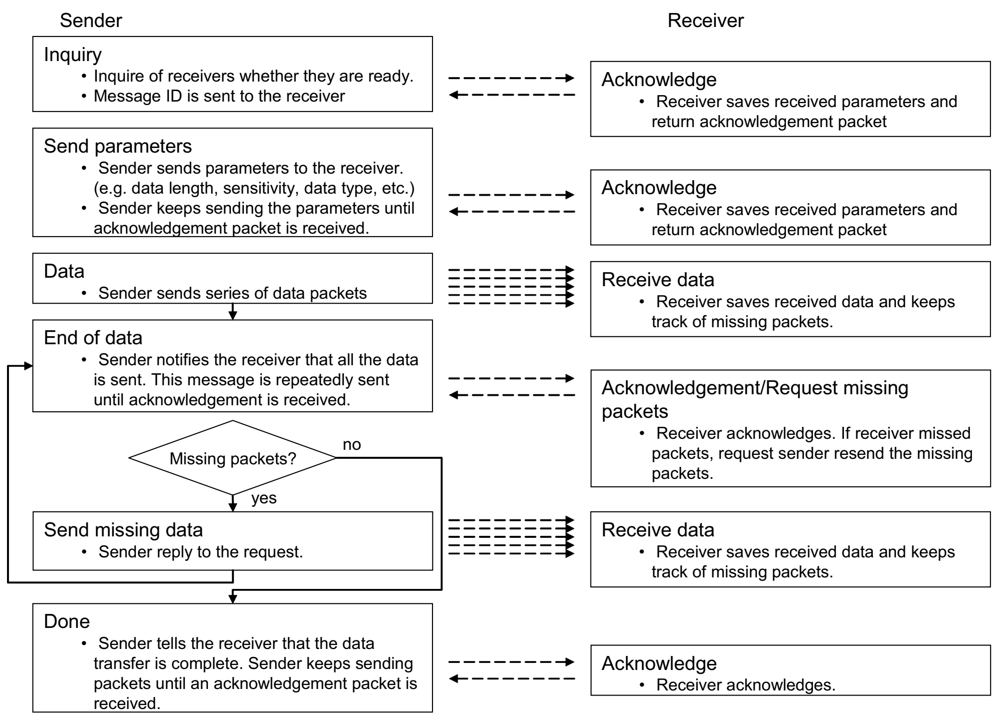

Sender
Receiver
Inquiry
• Inquire of receivers whether they are ready.
• Message ID is sent to the receiver
Acknowledge
• Receiver saves received parameters and return acknowledgement packet
Send parameters
• Sender sends parameters to the receiver. (e.g. data length, sensitivity, data type, etc.)
• Sender keeps sending the parameters until acknowledgement packet is received.
Acknowledge
• Receiver saves received parameters and return acknowledgement packet
Data
• Sender sends series of data packets
Receive data
• Receiver saves received data and keeps track of missing packets.
End of data
• Sender notifies the receiver that all the data is sent. This message is repeatedly sent until acknowledgement is received.
Acknowledgement/Request missing packets
• Receiver acknowledges. If receiver missed packets, request sender resend the missing packets.
Missing packets?
no
yes
Send missing data
• Sender reply to the request.
Receive data
• Receiver saves received data and keeps track of missing packets.
Done
• Sender tells the receiver that the data transfer is complete. Sender keeps sending packets until an acknowledgement packet is received.
Acknowledge
• Receiver acknowledges.

Figure 5.14. Simple block diagram of the reliable multicast protocol for long data

comment field of this packet is ‘f’ in hexadecimal. A maximum of eight receiver node IDs are sent in a single packet. When data is sent to more than eight nodes, multiple packets containing the receiver node IDs are broadcast. As is the case for unicast communication, the sender periodically sends these packets until the receivers reply.

Upon reception of this initial packet, a node check is conducted to determine whether the received node IDs contain its own node ID. If the node is not one of the destination nodes, this node simply disregards the packet. If the node is one of destination nodes, this node sends an acknowledgment to the sender and commits itself to this round of communication. Note that if multiple receivers reply to the sender at the same time, packet collision may take place; timing for replies to the sender needs to be scheduled appropriately. The order of node IDs in the packets with the comment ‘f’ is used to schedule the timing for replies. For example, the node corresponding to the eighth node ID in the packet with the ‘f’ comment waits 105 (= 15 ms x 7) ms before the acknowledgment is sent back to the sender. The order of and the waiting time for the reply are stored on each receiver; all of the following acknowledgment messages in this multicast protocol use this schedule. Optimization of the waiting interval, 15 ms, may result in faster communication, but such optimization is not pursued in this research.

Once the sender receives an acknowledgment from all of the receivers, the parameters such as data length, sensitivity, offset, and data type are broadcast. Although the broadcast transmission reaches all of the nodes in the neighborhood, only the engaged nodes process the received packet. The source ID, the destination ID (i.e., ID ‘0xffff’ which corresponds to broadcast address in TinyOS), and the message ID are used on the receiver to distinguish packets to be processed from other packets.

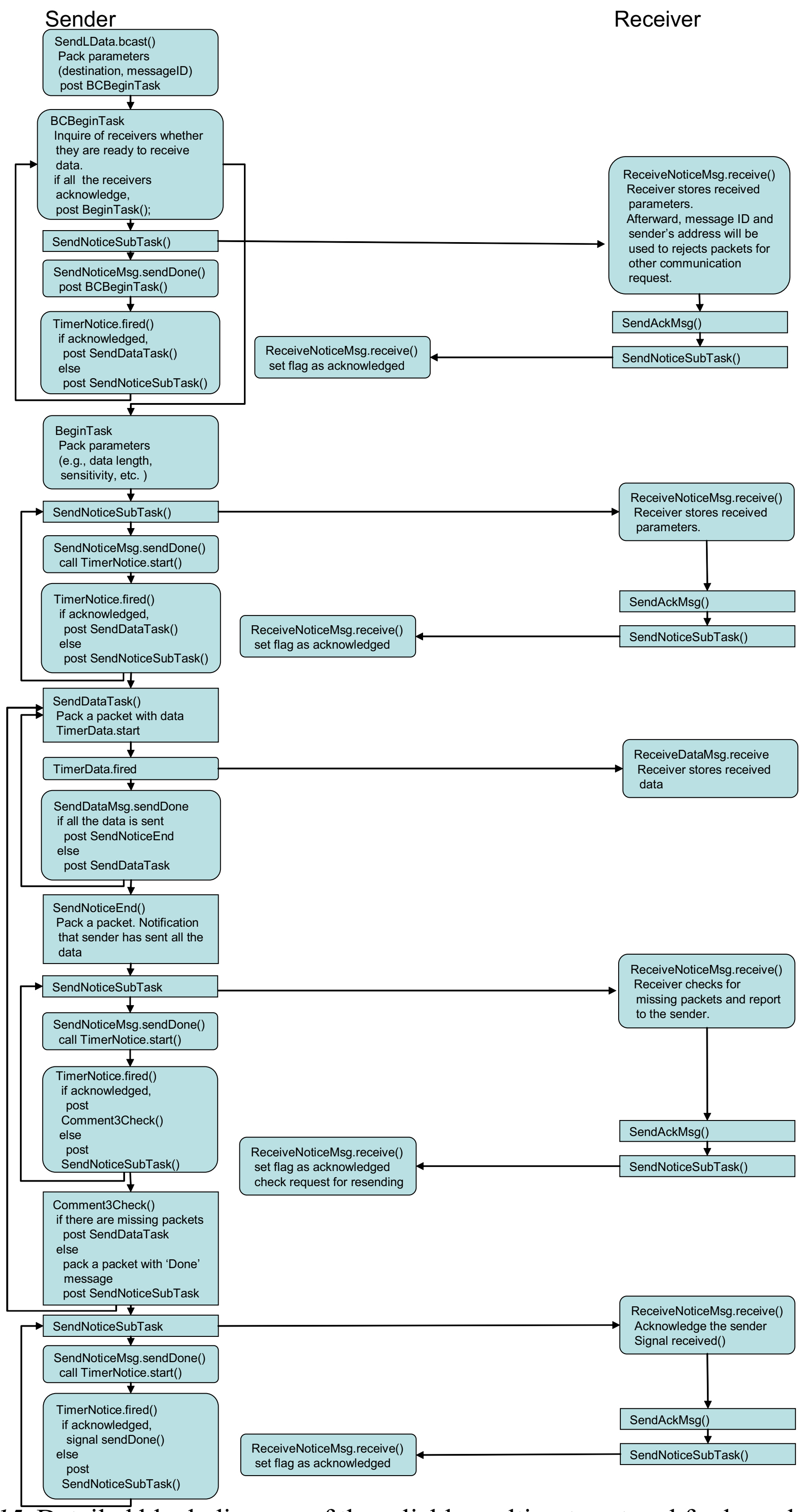

Sender
Receiver
SendLData.bcast() Pack parameters (destination, messageID) post BCBeginTask
BCBeginTask Inquire of receivers whether they are ready to receive data. if all the receivers acknowledge, post BeginTask();
SendNoticeSubTask()
SendNoticeMsg.sendDone() post BCBeginTask()
TimerNotice.fired() if acknowledged, post SendDataTask() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() Receiver stores received parameters. Afterward, message ID and sender's address will be used to rejects packets for other communication request.
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged
BeginTask Pack parameters (e.g., data length, sensitivity, etc.)
SendNoticeSubTask()
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, post SendDataTask() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() Receiver stores received parameters.
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged
SendDataTask() Pack a packet with data TimerData.start
TimerData.fired
ReceiveDataMsg.receive Receiver stores received data
SendDataMsg.sendDone if all the data is sent post SendNoticeEnd else post SendDataTask
SendNoticeEnd() Pack a packet. Notification that sender has sent all the data
SendNoticeSubTask
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, post Comment3Check() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() Receiver checks for missing packets and report to the sender.
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged check request for resending
Comment3Check() if there are missing packets post SendDataTask else pack a packet with 'Done' message post SendNoticeSubTask
SendNoticeSubTask
SendNoticeMsg.sendDone() call TimerNotice.start()
TimerNotice.fired() if acknowledged, signal sendDone() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() Acknowledge the sender Signal received()
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged

Figure 5.15. Detailed block diagram of the reliable multicast protocol for long data records

Another difference between unicast and multicast is that the sender needs to check the acknowledgment messages from all of the receivers. The sender keeps track of acknowledgments from the receivers using an array. One bit of the array corresponds to one receiver. Before the sender transmits a packet needing acknowledgement from the receiver, this array is initialized to zeroes. Upon reception of an acknowledgment from a receiver, the bit corresponding to that receiver is changed to one. While the size of this array is set so that up to 30 nodes can be receivers currently, parameters can be easily adjusted to accommodate a larger number of receivers at the expense of memory space required for the array.

Another point to consider for this multicast communication protocol is the way that requests for retransmission are handled. After sending the end-of-data notice to the receivers, the sender waits for a reply from the respective receivers. Within the assigned time slot, each receiver reports to the sender; a packet containing missing packet numbers is sent back. After the time assigned for all the nodes to reply has passed, the sender broadcasts the requested packets. At the end of broadcast of these requested packets, a packet with the comment ‘2’ is sent, asking for packets still missing.

The end of one round is difficult to judge, as explained in the unicast reliable communication protocol section. A similar solution is implemented for multicast reliable communication. In addition to the message and source IDs, this multicast protocol needs to store the time to wait for before replying with an acknowledgment message for the last 15 rounds of communications.

2. Communication protocol for short messages

Communication protocols suitable to transfer a single packet reliably are also needed. SHM applications involve sending and receiving many commands, each of which fits in one packet. These commands need to be delivered reliably. If a packet containing a command to start sensing is lost, the destination node does not start sensing. If loss of a packet is not detected, the sender assumes the destination nodes has performed tasks based on the command. The current state of a smart sensor node is difficult to estimate without reliable communication. Ensuring the performance of SHM systems without reliable command delivery is extremely complex if not impossible. The reliable communication protocol developed for long data records is, however, not efficient for single packet transfer. Unicast and multicast reliable communication protocols suitable for single-packet messages are developed in this section.

This protocol is again similar to the ARQ protocol. Because only one packet is sent, an acknowledgment is returned to the sender for each packet, as is the case for the stop-and-wait protocol. The protocol is designed to send commands and parameters in 8- and 16-bit integers. One packet can hold nine 16-bit integers and one 8-bit integer in addition to the parameters necessary for reliable communication.

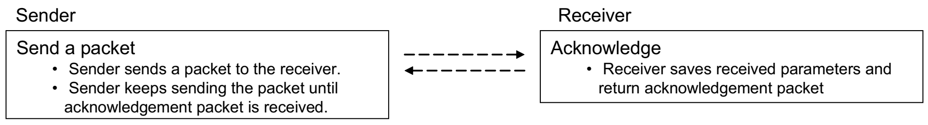

Sender
Send a packet
• Sender sends a packet to the receiver.
• Sender keeps sending the packet until acknowledgement packet is received.
Receiver
Acknowledge
• Receiver saves received parameters and return acknowledgement packet

Figure 5.16. Simple block diagram of the reliable unicast protocol for short messages.

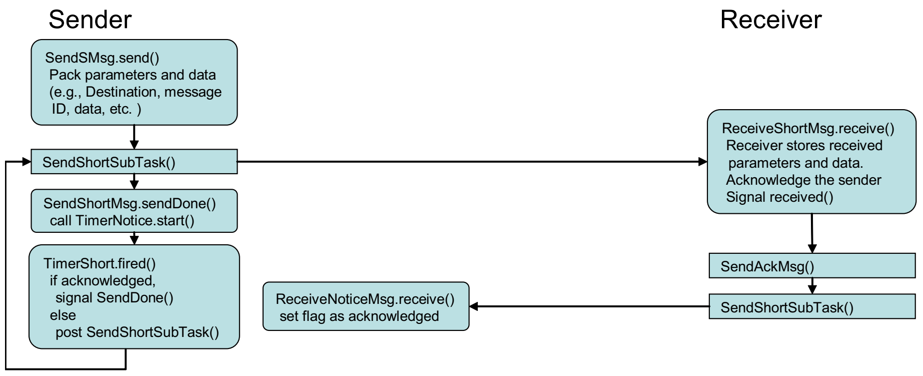


Figure 5.17. Detailed block diagram of the reliable unicast protocol for short messages.

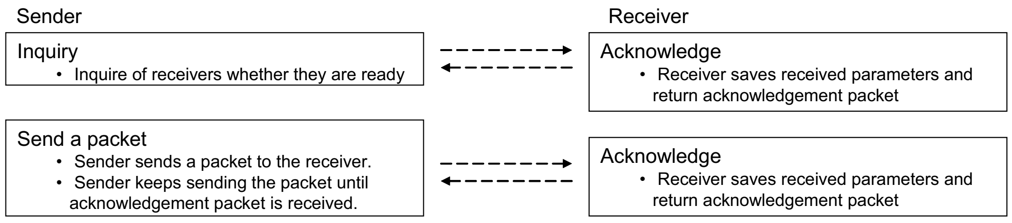

Sender
Inquiry
• Inquire of receivers whether they are ready
Send a packet
• Sender sends a packet to the receiver.
• Sender keeps sending the packet until acknowledgement packet is received.
Receiver
Acknowledge
• Receiver saves received parameters and return acknowledgement packet
Acknowledge
• Receiver saves received parameters and return acknowledgement packet

Figure 5.18. Simple block diagram of the reliable multicast protocol for short messages.

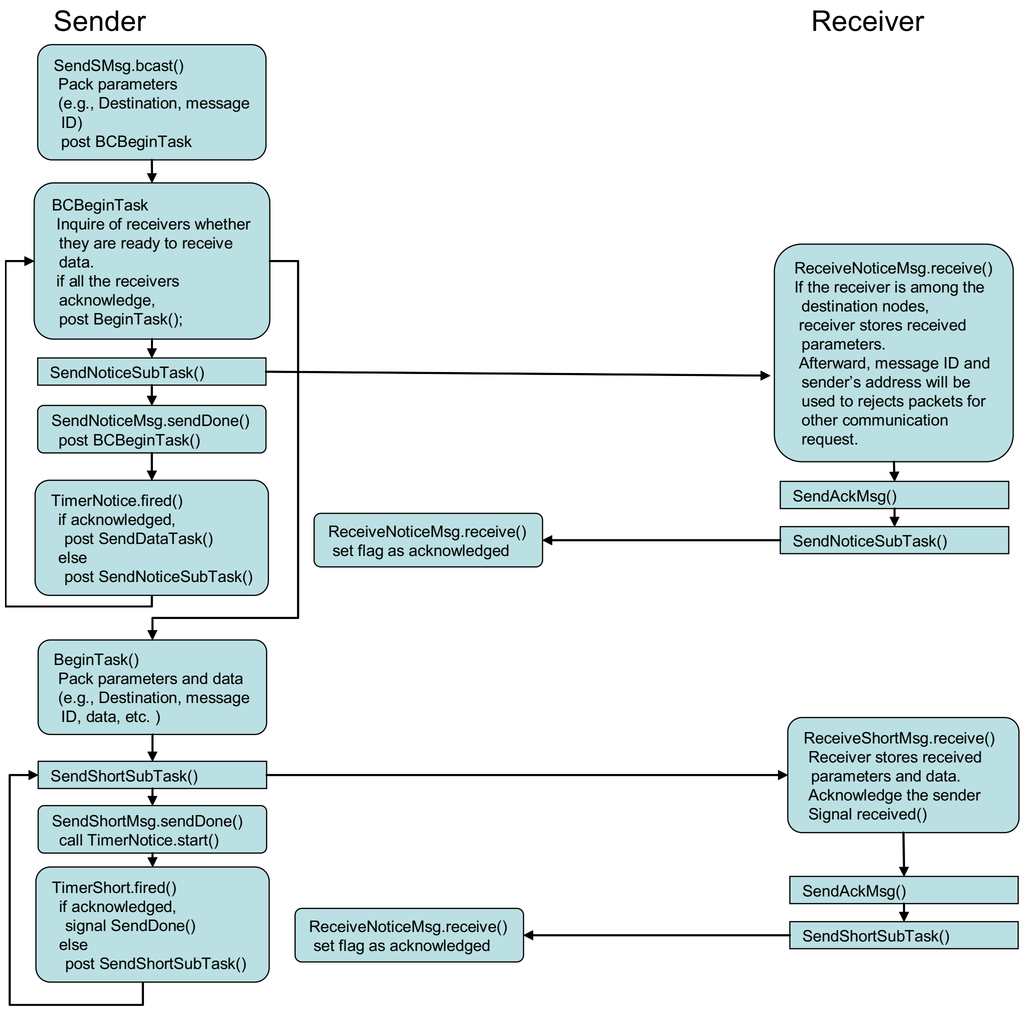
Unicast

The sender transmits a packet, including the destination ID, source ID, message ID, parameter, and commands. Until this packet is acknowledged, the sender continues to transmit this packet. The receiver extracts information from the packet and sends an acknowledgment back to the sender (see Figures 5.16 and 5.17)

The end of one round is difficult to judge, as explained in the reliable unicast protocol for long data records. The same solution applies to short messages. The source and message IDs for the last 15 rounds of communication are stored so that acknowledgment can be sent even after the receiver is disengaged from this round of communication.

Multicast

The sender first transmits a packet including the receivers' node IDs, as was the case for the multicast communication protocol for long data records. Once all of the receivers return an acknowledgment to the sender, the sender transmits a packet containing parameters and commands. The receivers also acknowledge this packet. To judge the end



Sender
Receiver
SendSMsg.bcast() Pack parameters (e.g., Destination, message ID) post BCBeginTask
BCBeginTask Inquire of receivers whether they are ready to receive data. if all the receivers acknowledge, post BeginTask();
SendNoticeSubTask()
SendNoticeMsg.sendDone() post BCBeginTask()
TimerNotice.fired() if acknowledged, post SendDataTask() else post SendNoticeSubTask()
ReceiveNoticeMsg.receive() If the receiver is among the destination nodes, receiver stores received parameters. Afterward, message ID and sender's address will be used to rejects packets for other communication request.
SendAckMsg()
SendNoticeSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged
BeginTask() Pack parameters and data (e.g., Destination, message ID, data, etc.)
SendShortSubTask()
SendShortMsg.sendDone() call TimerNotice.start()
TimerShort.fired() if acknowledged, signal SendDone() else post SendShortSubTask()
ReceiveShortMsg.receive() Receiver stores received parameters and data. Acknowledge the sender Signal received()
SendAckMsg()
SendShortSubTask()
ReceiveNoticeMsg.receive() set flag as acknowledged

Figure 5.19. Detailed block diagram of the reliable multicast protocol for short messages.

of communication, the receiver keeps the source ID, message ID, and waiting time of the last 15 rounds of communication (see Figures 5.18 and 5.19).

5.3 Synchronized sensing

Time synchronization error in a smart sensor network can cause inaccuracy in SHM applications. Time synchronization is a middleware service common to smart sensor applications and has been widely investigated. Each smart sensor has its own local clock, which is not synchronized initially with other sensor nodes. By communicating with surrounding nodes, smart sensors can assess relative differences among their local clocks. For example, Mica2 motes employing TPSN are reported to synchronize with each other to an accuracy of 50 μsec; different algorithms and hardware resources may result in different precision. Whereas time synchronization protocols have been intensively studied, requirements for synchronization from an application perspective have not been clearly addressed. The effect of time synchronization on SHM applications is first studied. As stated earlier, spectral density and correlation function estimation are oftentimes utilized in SHM applications followed by modal analysis. Nagayama et al. (2007) has discussed the effects of synchronization errors on these estimates. Time synchronization accuracy realized on the Imote2, the smart sensor platform employed in this research, is then estimated and evaluated for the SHM applications. Because time synchronization among smart sensors does not necessarily offer synchronized measurement signals, issues critical to synchronized sensing are then investigated. Finally, synchronized sensing is realized utilizing resampling (Nagayama et al., 2006a; Spencer & Nagayama, 2006).

5.3.1 Time synchronization effect on SHM applications

Consider the signal $x(\bar{t})$ from a smart sensor in the local clock coordinate $\bar{t}$. This signal can be written in terms of the reference (or global) clock t as

$$x(\bar{t}) = x((1+\alpha)t - \delta t) \tag{5.4}$$

where δt is the initial time synchronization error and α is the clock drift rate. In the frequency domain, this relationship is expressed as

$$X(\bar{\omega}) = \frac{1}{1+\alpha}\exp\left(-i\omega\frac{\delta t}{1+\alpha}\right)X\left(\frac{\omega}{1+\alpha}\right) \tag{5.5}$$

where $X(\omega)$ and $X(\bar{\omega})$ are the Fourier transform of $x(t)$ and $x(\bar{t})$, respectively. In the following derivation, α is assumed to be zero.

The effect of time synchronization errors on modal parameters, such as the natural frequency, damping ratio, and mode shape, are examined. Though only output measurements can be obtained during most of the civil infrastructure monitoring, for completeness, the effect of time synchronization error is investigated both for the input-output measurement case and the output-only measurement case.

When both input force and output structural responses are measured, transfer functions from input to output signals are first estimated and then modal analysis follows. Therefore, the effect of time synchronization error on transfer function estimates is studied.

Consider a structure for which the equation of motion is written as

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{f}(t) \tag{5.6}$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are mass, damping, and stiffness matrices, respectively. $\mathbf{x}(t)$, $\dot{\mathbf{x}}(t)$, $\ddot{\mathbf{x}}(t)$, and $\mathbf{f}(t)$ are the displacement, velocity, acceleration, and input force vectors, respectively. By taking the Fourier transform of Eq. (5.6), the displacement vector can be written as

$$\begin{aligned}\mathbf{X}(\omega) &= (-\mathbf{M}\omega^2 + \mathbf{C}i\omega + \mathbf{K})^{-1}\mathbf{F}(\omega)\\ &= \mathbf{T}_{\mathbf{FX}}(\omega)\mathbf{F}(\omega)\end{aligned} \tag{5.7}$$

where $\mathbf{X}(\omega)$ and $\mathbf{F}(\omega)$ are the Fourier transforms of $\mathbf{x}(t)$ and $\mathbf{f}(t)$, respectively. $\mathbf{T_{FX}}(\omega)$ is the transfer function from the input force to the displacement. The *i*-th element of $\mathbf{x}(t)$, $x_i(t)$ with time synchronization error of δt_{x_i} is modeled as $x_i(t\text{-}\delta t_{x_i})$. By transforming to the frequency domain, the measured displacement vector of size *n*, $\mathbf{x}(\bar{t})$ with time delay δt_i, $i=1,2,...,n$ is written in frequency domain as

$$\mathbf{X}(\bar{\omega}) = \begin{bmatrix} \exp(-i\omega\delta t_1) & 0 & \dots & 0 \\ 0 & \exp(-i\omega\delta t_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \exp(-i\omega\delta t_n) \end{bmatrix} \mathbf{X}(\omega) \tag{5.8}$$

A similar relation holds for the measured force $\mathbf{f}(\bar{t})$. Therefore, the transfer function from the input force to the displacement with synchronization error, $\mathbf{T_{FX}}(\bar{\omega})$ is expressed as in Eq. (5.9),

$$\mathbf{T_{FX}}(\bar{\omega}) = \begin{bmatrix} \exp(-i\omega\delta t_{x_1}) & 0 & \dots & 0 \\ 0 & \exp(-i\omega\delta t_{x_2}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \exp(-i\omega\delta t_{x_n}) \end{bmatrix} \tag{5.9}$$

$$\mathbf{T_{FX}}(\omega) \begin{bmatrix} \exp(i\omega\delta t_{f_1}) & 0 & \dots & 0 \\ 0 & \exp(i\omega\delta t_{f_2}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \exp(i\omega\delta t_{f_m}) \end{bmatrix}$$

where δt_{f_i} is the time synchronization error for the *i*-th force measurement. Eq. (5.9) shows that the time synchronization error does not affect the denominator of the transfer function; natural frequencies and damping ratios determined only by the denominator are immune to time synchronization error. The numerator, however, is multiplied by complex values of unit magnitude, resulting in phase errors in the mode shapes.

When only output structural responses are measured, correlation functions between the output measurements can be used to determine modal parameters, as explained in section 6.1. Therefore, the effect of time synchronization on correlation functions and the associated modal parameters is investigated.

Consider a cross correlation function $R_{x_i x_{ref}}(\tau)$ between the stationary random response $x_i(t)$ at location *i* and reference signal $x_{ref}(t)$. $R_{x_i x_{ref}}(\tau)$ satisfies the equation of motion for free vibration given by

$$\mathbf{M}\ddot{\mathbf{R}}_{\mathbf{x}x_{ref}}(\tau) + \mathbf{C}\dot{\mathbf{R}}_{\mathbf{x}x_{ref}}(\tau) + \mathbf{K}\mathbf{R}_{\mathbf{x}x_{ref}}(\tau) = 0, \tau \geq 0 \tag{5.10}$$

where $\mathbf{R}_{\mathbf{x}x_{ref}}$ is a vector consisting of $R_{x_i x_{ref}}(\tau)$. When the sensor node at location *i* has a time synchronization error of δt_{x_i} relative to the reference node, the correlation function, $\bar{R}_{x_i x_{ref}}(\tau)$ can be written as

$$\bar{R}_{x_i x_{ref}}(\tau) = \mathrm{E}[x_i(t-\delta t_{x_i}+\tau)x_{ref}(t)] = R_{x_i x_{ref}}(t-\delta t_{x_i}) \qquad (5.11)$$

where E[ ] is the expectation operator. If δt_{x_i} is positive, the correlation function for the interval (0, δt_{x_i}) does not have the same characteristics as the succeeding signal. This portion of the cross correlation function will correspond to negative damping; therefore, the beginning portion of the signal needs to be removed. When δt_{x_i} is unknown, a segment corresponding to the maximum possible time synchronization error, $\delta t_{x_{max}}$, is truncated from the correlation function.

The correlation functions, each having independent time synchronization errors, do not satisfy the equation of motion, Eq. (5.10). The correlation functions after the truncation, $R_{x_i x_{ref}}'(\tau)$, however, can be decomposed into modal components as follows:

$$\begin{aligned}
R_{x_i x_{ref}}'(\tau) &= \Phi'\Lambda\mathbf{A} \\
\Phi' &= \begin{bmatrix}\varphi_1 & \varphi_2 & \cdots & \varphi_{2n}\end{bmatrix} \\
&= \begin{bmatrix}
\phi_{11}\exp(-\lambda_1\delta t_{x_1}) & \phi_{21}\exp(-\lambda_2\delta t_{x_1}) & \cdots & \phi_{2n1}\exp(-\lambda_{2n}\delta t_{x_1}) \\
\phi_{12}\exp(-\lambda_1\delta t_{x_2}) & \phi_{22}\exp(-\lambda_2\delta t_{x_2}) & \cdots & \phi_{2n2}\exp(-\lambda_{2n}\delta t_{x_2}) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_{1m}\exp(-\lambda_1\delta t_{x_m}) & \phi_{2m}\exp(-\lambda_2\delta t_{x_m}) & \cdots & \phi_{2nm}\exp(-\lambda_{2n}\delta t_{x_m})
\end{bmatrix} \\
\Lambda &= diag(\begin{bmatrix}\exp(\lambda_1\tau) & \exp(\lambda_2\tau) & \cdots & \exp(\lambda_{2n}\tau)\end{bmatrix}) \\
\lambda_i &= h_i\omega_i + j\omega_i\sqrt{1-h_i^2} \\
\mathbf{A} &= diag(\begin{bmatrix}a_1 & a_2 & \cdots & a_{2n}\end{bmatrix})
\end{aligned} \qquad (5.12)$$

where ϕ_{ij} is *j*-th element of *i*-th mode shape, h_i and ω_i are *i*-th modal damping ratio and modal natural frequency, respectively, and a_i is a factor accounting for the relative contribution of the *i*-th mode in the correlation function matrix. Modal analysis techniques such as ERA can be used to identify these modal parameters. As Eq. (5.12) shows, the natural frequencies and damping ratios remain the same. The observed mode shapes Φ' are, however, different from the original mode shapes; changes in mode shape amplitude are negligible due to small h_i and t_{x_i}; however, phase shifts in the mode shapes can be substantial. Mode shape phases can indicate structural damage and are important modal characteristics from an SHM perspective. The requirements on time synchronization accuracy for modal analysis need to be assessed mainly from the viewpoint of the mode shape phase.

Though the effects of time synchronization error on transfer functions and correlation functions are shown, the estimates calculated from a finite length of measured data are not exactly the same as these functions. This issue is not theoretically pursued herein. However, numerical simulation conducted to date using finite length simulation data records supports the relations given in Eqs (5.9) and (5.12). Thus, requirements on time synchronization errors are concluded to be dictated primarily by their effect on mode shape phase.

5.3.2 Estimation on time synchronization error

The accuracy of the time synchronization protocol implemented on the Imote2 is evaluated. The Flooding Time Synchronization Protocol (FTSP) is implemented on the Imote2 under the advice of Kirill Mechitov at the University of Illinois at Urbana-Champaign and Intel Corporation. This protocol is first briefly reviewed.

FTSP utilizes time stamps on the sender and receivers. A beacon node broadcasts a packet to the other nodes. At the end of the packet, a time stamp, t_{send}, is appended just before the packet is transmitted from the RF device on the beacon. Upon reception of the packet, the receivers stamp the time, $t_{receive}$, from their own local clocks. The delivery time, $t_{delivery}$, between these two events includes interrupt handling time, encoding time, and decoding time. $t_{delivery}$ is usually not small enough to be ignored; the variance of $t_{delivery}$ over time is usually small. $t_{delivery}$ can be estimated in several ways. An oscilloscope connected to both nodes and on-board clocks can keep track of the communication time stamp. In this research, $t_{delivery}$ is first assumed to be zero and then adjusted so that Imote2s placed on the same shake table give synchronized acceleration data. The phases of the transfer functions among Imote2 signals should be constant with zero phase over wide frequency range if these nodes are synchronized. $t_{delivery}$ is determined to give a constant phase of zero. The details of this adjustment are described in section 5.3.4. The offset between the local clock on the receiver and the reference clock on the sender is determined as $t_{receive} - t_{send} - t_{delivery}$. This offset is subtracted from the local time when global time stamps are requested afterward.

To evaluate time synchronization error, a group of nine Imote2s are programmed as follows. The beacon node transmits a beacon signal every 4 seconds. The other eight nodes estimate the global time using the beacon packet. Time synchronization is, thus, performed. Two seconds after the beacon signal, the beacon node sends the second packet, requesting replies. The receivers get time stamps on reception of this packet and convert them to a global time stamp using the offset estimated 2 seconds before. These time stamps are subject to two error sources: First, time synchronization error, and second, delay in time stamping upon reception of the second packet. The receivers take turns to report back these time stamps. This procedure is repeated more than 300 times. These time stamps from the eight nodes are compared. Figure 5.20 shows the difference in the respective global time stamps using one of the eight nodes as a reference. The time synchronization error seems to be less than 10 μs for most of the time. Scattered peaks may indicate large synchronization error. Note that the time synchronization error is one

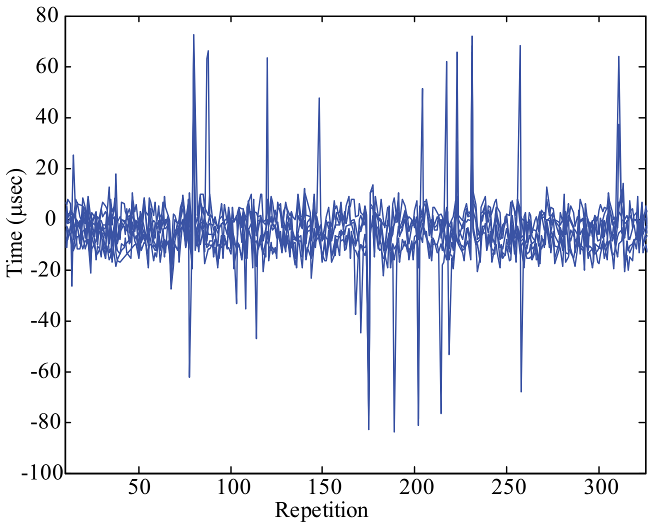


Figure 5.20. Time synchronization error estimation.

of the two above-mentioned factors explaining the error in the time stamps. This figure indicates that an upper-bound estimate of time synchronization error is 80 μs.

The time synchronization error estimated above is considered small for SHM applications. The delay of 10 μsec corresponds to a 0.072-degree phase error for a mode at 20 Hz. Even at 100 Hz, the corresponding phase error is only 0.36 degree.

While a global clock estimates 2 seconds after a beacon signal is found to be accurate, local clocks may drift over time. Large clock drift necessitates frequent time synchronization to maintain certain accuracy. The clock drift in the Imote2 is estimated next.

The same approach is utilized to estimate clock drift. Upon reception of the second packet, which requests replies, the receivers return to the sender their offsets to estimate the global time, instead of global time stamps. If clocks on nodes are ticking at exactly the same rate, the offsets should be constant over a long time. This experiment, however, did not show constant offsets. Figure 5.21 shows the offsets of nine receiver nodes. One of them stopped responding around 45 seconds, exhibiting a short line on the figure. This figure shows that the drift is quite constant in time. The maximum clock drift among this set of Imote2 nodes is estimated to be around 50 μs per second. Note that this estimate from the nine nodes is not the upper limit of clock drift because of the small sample size. This drift is small but not negligible if measurement takes a long time. For example, after 200 second measurement, the time synchronization error may become as large as 10 ms.

One solution to address this clock drift problem is frequent time synchronization. Time synchronization is performed often before the effect of clock drift accumulates and results in large time synchronization error. However, frequent time synchronization is not always feasible. When other tasks are running, such as sensing, time synchronization may not perform well. Time synchronization requires precise time stamping as explained earlier. Moreover, sensing requires precise timing and needs higher priority in execution. Scheduling more than one high priority tasks is challenging, especially for an operating system such as TinyOS which has no support for real-time control. If the time

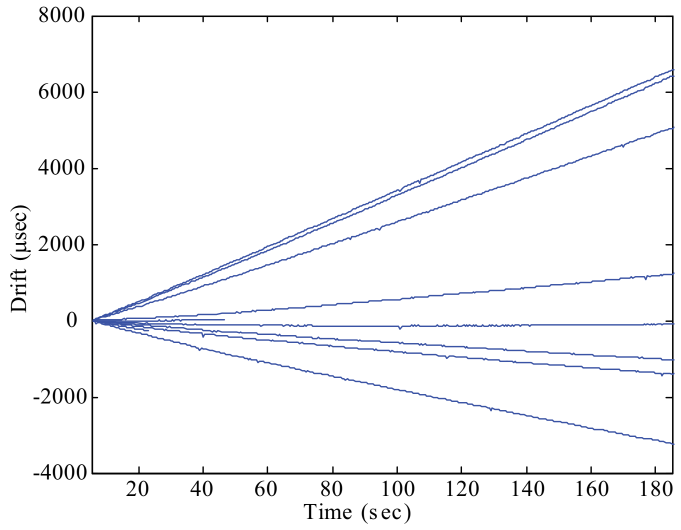


Figure 5.21. Drift estimation.

synchronization interval is shorter than the sensing duration, another solution needs to be sought to avoid scheduling both time synchronization and sensing tasks.

Alternatively, a clock's drift rate can be compensated directly. The slopes of the lines in Figure 5.21 are nearly constant and provide an estimate of the clock drift rate to be compensated. If time synchronization offset values can be observed for a certain amount of time, the slope can be estimated using a least-squares approach.

5.3.3 Issues toward synchronized sensing

Accurate synchronization of local clocks on Imote2s does not guarantee that measured signals are synchronized. Measurement timing cannot necessarily be controlled based on the global time.

Sensing on Imote2s is performed in the following way. A sensing application posts a task to prepare for sensing. Parameters such as the sampling frequency and the total number of data points are passed to the driver. Once the sensor driver is ready, sensing starts. The sensing task continues running until the predetermined amount of data is acquired. During this sensing, the acquired data points are first stored in a buffer. Every time the buffer is filled, the driver passes the data to the sensing application. This block of data is supplied with a local time stamp marked when the last data point of the block is acquired. The clock used for the time stamp runs at 3.25 MHz. If time synchronization is performed prior to sensing, the offset between the global and local times can be utilized to convert the local time stamp to a global time stamp where needed. The data and time stamps passed to a sensing application are copied to arrays, and the buffer is returned to the driver to be used for the next block of data. The size of the buffer was set to keep 110 data points. When 1100 data points are acquired, for example, the buffer is filled ten times; everytime the buffer is filled, its contents are passed to the application.

Difficulties encountered in realizing synchronized sensing using Imote2s are summarized in the following paragraphs.

1. Failure in sensing

Imote2s do not always succeed in sensing. When a node fails to start sensing, no acceleration record is stored on this node. Processing data without recognizing such failures results in large errors in the final outcome. Failures in sensing need to be detected so that data from the associated nodes is not processed together with the other data.

2. Uncertainty in start-up time

Starting up sensing tasks at all of the Imote2 nodes simultaneously is challenging. Even when the commands to start sensing are queued at exactly the same time, the execution timing of the commands is different on each node. Thus, measured signals are not necessarily synchronized to each other.

TinyOS has only two types of threads of execution: tasks and hardware event handlers, leaving users little control to assign priority to commands; if sensing is queued as a task, this task is executed after all the tasks in the front of the queue are completed. As such, waiting time cannot be predicted. If the command is invoked in a hardware interrupt as an asynchronous command, this command is executed immediately unless other hardware interrupts interfere. However, invocation of commands as a hardware interrupt from a clock firing at very high frequency is not practical; firing the timer at a frequency corresponding to the synchronization accuracy, tens of microseconds, is impossible.

In addition, the warm-up time for sensing devices after the invocation of the start command is not constant. Even if the commands are invoked at the same time, sensing will not start simultaneously.

3. Difference in sampling rate among sensor nodes

The sampling frequency of the accelerometer on the available Imote2 sensor boards has nonnegligible random error. According to the data sheet of the accelerometer, the sampling frequency may differ from the nominal value by at most 10 percent (STMicroelectronics, 2007). Such variation was observed when 14 Imote2 sensor boards

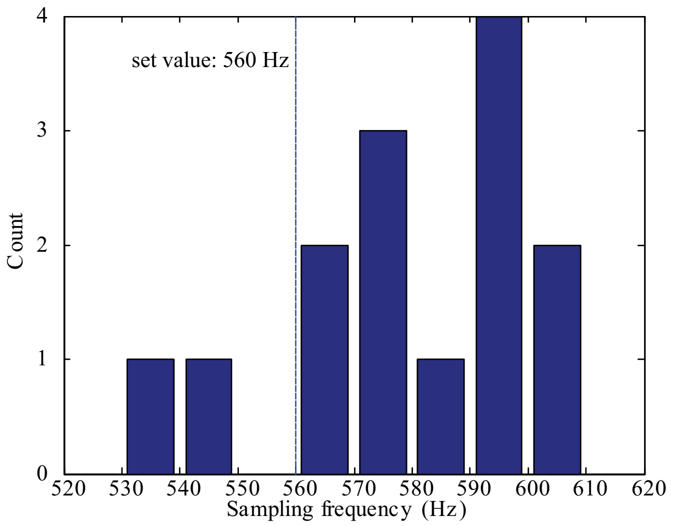


Figure 5.22. Sampling frequencies of 14 sensor boards.
(Frequency calibration was conducted by Jennifer A. Rice.)

were calibrated on a shake table (see Figure 5.22). Differences in the sampling frequencies among the sensor nodes result in inaccurate estimation of structural properties unless appropriate post processing is performed. If signals from sensors with nonuniform sampling frequency are used for modal analysis, one physical mode may be identified as several modes spread around the true natural frequency (Nagayama et al., 2006a).

4. Fluctuation in sampling frequency over time

A nonconstant sampling rate was observed with the Imote2 sensor boards, which if not addressed, results in a seriously degraded acceleration measurement. The Imote2 receives the digital acceleration signal once a block of data is available from the accelerometer. The block size is set at 110 data points. The Imote2 puts a time stamp on the data once a block is available. By comparing differences between two consecutive time stamps, the sampling frequency of the accelerometer is estimated. The difference between the time stamps, shown in Figure 5.23, provides an indication of the variation in

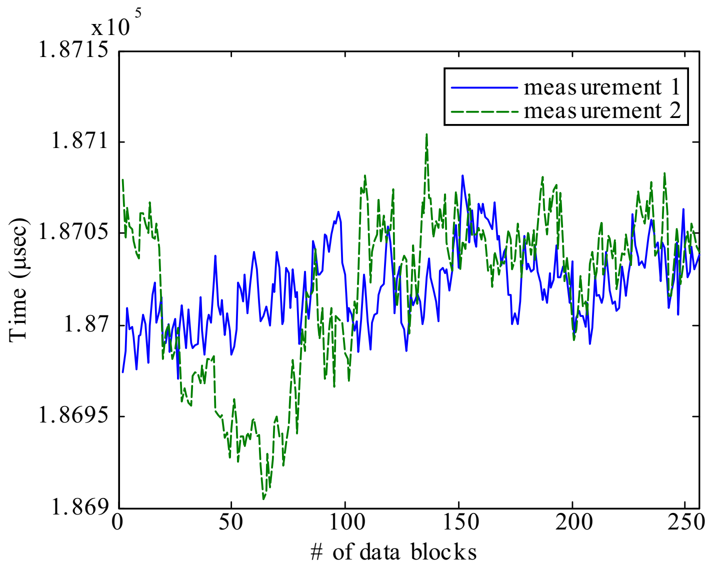


Figure 5.23. Variation in the sampling frequency over time.

the sampling frequency. The difference in two consecutive timestamps fluctuates by about 0.1 percent. Though imperfect time stamping on Imote2 is a possible source of the apparent nonconstant sampling rate, fluctuation with a nonzero average values suggests that the variable sampling frequency as a credible cause of the phenomenon. With this fluctuation, measurement signals may suffer from a large synchronization error.

5.3.4 Realization of synchronized sensing

The observed problems discussed previously are addressed using the smart sensor's computational capabilities. Failure during sensing is detected and sensing is repeated until success is achieved. The other three issues are dealt with concurrently by resampling the measured time histories based on time stamps at the end of each block of data. These two approaches realize synchronized sensing and are discussed in this section.

1. Detection of sensing failure and repetition of sensing

To detect sensing failure, a timer on the Imote2 is utilized. This timer is set when sensing is initiated; the timer is scheduled to be fired after the planned total measurement time has passed. If sensing is successfully completed, this timer is stopped at the end of sensing. Therefore, the timer never fires unless sensing fails during measurement. If the timer fires, a flag is used to mark sensing failure. After a predetermined time has passed, the manager sensor inquires of the other nodes and itself whether sensing has failed. If a flag indicating sensing failure is found, the memory space storing acquired data is reinitialized, and a task to restart sensing is posted at all of the nodes.

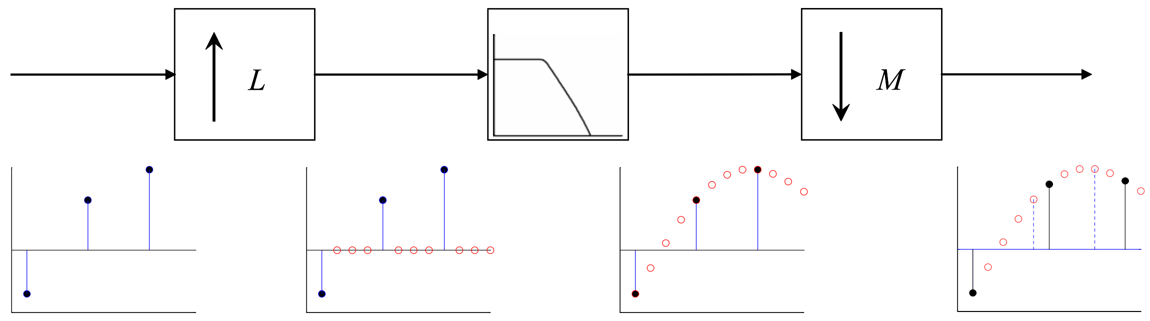


Figure 5.24. The basic idea of resampling.

2. Resampling-based approach

Resampling based on the global time stamps addresses three of the problems previously cited: (a) uncertainty in start-up time, (b) difference in sampling rate among sensor nodes, and (c) time fluctuation in the sampling frequency. The basics of resampling and polyphase implementation of resampling are first reviewed. The resampling approach is then modified to achieve a sampling rate conversion by a noninteger factor. Finally, this proposed resampling method is combined with time stamps of measured data to address concurrently the three issues.

Resampling

Resampling by a rational factor is performed by a combination of interpolation, filtering, and decimation. Consider the case in which the ratio of the sampling rate before and after resampling is expressed as an rational factor, L/M. The signal is first upsampled by a factor of L. Then the signal is downsampled by a factor of M. Before downsampling, a lowpass filter is applied to eliminate aliasing and unnecessary high-frequency and aliasing components (see Figure 5.24).

During upsampling, the original signal $x[n]$ is interpolated by L-1 zeroes as in Eq. (5.13),

$$y[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \mathit{otherwise} \end{cases} \tag{5.13}$$

where $y[n]$ is the upsampled signal. In the frequency domain, insertion of zeroes creates scaled mirror images of the original signal in the frequency range between the new and old Nyquist frequencies. A discrete-time lowpass filter with a cutoff frequency, π/L, is applied so that all of these images except for the one corresponding to the original signal are eliminated. To scale properly, the gain of the filter is set to be L.

The signal is then downsampled by a factor of M as in Eq. (5.14).

$$z[n] = y[nM] \tag{5.14}$$

Before this decimation is applied, all of the frequency components above the new Nyquist frequency need to be eliminated. A discrete-time, lowpass filter with a cutoff frequency π/M and gain 1 is applied.

The lowpass filter applied in the downsampling can be combined with the one in the upsampling process. The cutoff frequency of the filter is set to be the smaller value of π/L and π/M. The gain is L. This filtering process is reviewed more in more detail in the subsequent paragraphs.

Numerical filters of various types can be represented as in the following equation,

$$y[n] = \sum_{k=0}^{N_b} b[k]x[n-k] - \sum_{k=1}^{N_a} a[k]y[n-k] \tag{5.15}$$

where $x[n]$ is the input to the filter, $y[n]$ is the output from the filter, $a[k]$ and $b[k]$ are filter coefficients for outputs and inputs, respectively. N_a and N_b represent the numbers of filter coefficients, $a[k]$ and $b[k]$. These filters can be classified as either Finite Impulse Response (FIR) filters or Infinite Impulse Response (IIR) filters. An FIR filter has nonzero coefficients corresponding only to the inputs. In other words, N_a is zero for an FIR filter. An IIR filter has one or more filter coefficients corresponding to the outputs.

FIR and IIR filters have their own advantages over the other. FIR filters are always stable no matter what coefficients are chosen. Another advantage of an FIR filter is its linear phase characteristic. The delay introduced by an FIR filter is constant in frequency. On the other hand, IIR filters with a given performance can be designed using fewer coefficients, and, thus, are usually less computationally expensive.

One of the possible error sources of this resampling process is imperfect filtering. A perfect filter, which has a unity gain in the passband and a zero in the stopband, needs an infinite number of filter coefficients. With a finite number of filter coefficients, passband and stopband ripples cannot be zero. A filter design with 0.1 to 2 percent ripple is frequently used. A filter needs to be designed considering these filter characteristics. Figure 5.25 shows signals before and after filtering. A signal analytically defined as a combination of sinusoidal waves is sampled at three slightly different sampling frequencies. Two of the signals are then resampled at the sampling frequency of the other signal. As can be seen from Figure 5.25, after resampling, the three signals are almost identical. These signals are, however, not exactly the same due to the imperfect filtering. Though this signal distortion during filtering is preferably suppressed, this resampling process is not the only cause of such distortion. AA filtering and digital filtering also use imperfect filters. The filter in the resampling process needs to be designed so that the filter does not severely degrade signals as compared with other filters.

The resampling process is considered to be extremely challenging if the upsampling factor, L, is large. This issue is explained herein with examples. When a signal sampled at 100 Hz is resampled at 150 Hz, the rational factor, L/M, is 3/2. The original signal is upsampled by a factor of 3. A lowpass filter with a cutoff frequency of $\pi/3$ can be easily designed with a reasonable number of filter coefficients. Note that the cutoff frequency of

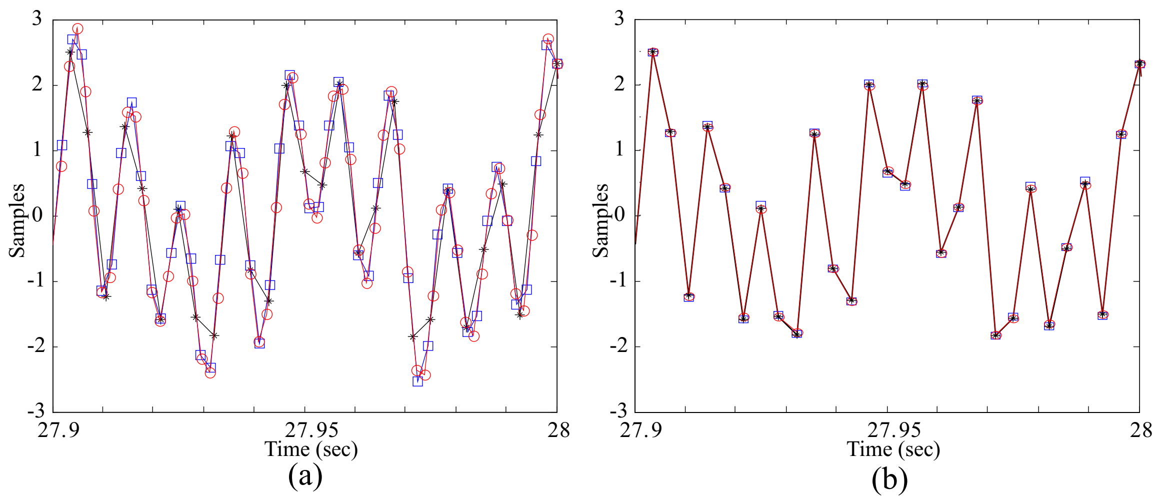


Figure 5.25. Signals (a) before and (b) after resampling.

the filter is expressed in radian normalized to the sampling frequency of an upsampled signal. This filter is applied to the upsampled signal, which is three times longer than the original one, and then downsampled by a factor of 2. When a signal sampled at 101 Hz is resampled at 150 Hz, on the other hand, the rational factor, L/M, is 150/101. Upsampling by a factor of 101 greatly increases the data size. A lowpass filter with a cutoff frequency of $\pi/150$ requires a large number of filter coefficients. If such a filter is applied to the upsampled signal, the upsampled signal is 101 times longer than the original one. Direct implementation of such resampling on resource-limited smart sensors is intractable.

Polyphase implementation

An FIR filter can be computationally much more inexpensive as a filter for resampling if the polyphase implementation is employed. The polyphase implementation leverages knowledge that upsampling involves inserting many zeroes and that an FIR filter does not need to calculate the output at all of the upsampled data points. This implementation of resampling is explained in this section mainly in the time domain because the extension of the method to achieve synchronized sensing is pursued using a time domain analysis. Oppenheim et al. (1999) provided a detailed description of the polyphase implementation in the Z-domain.

Upsampling and lowpass filtering with an FIR filter can be written in the following manner:

$$y[n] = \sum_{k=0}^{N-1} h[k]v[n-k]$$
$$v[l] = \begin{cases} x[m], & l = Lm,\ m = 0,\ \pm 1, \pm 2, \ldots \\ 0, & otherwise \end{cases} \tag{5.16}$$

where $x[n]$ is the original signal, $v[l]$ is the upsampled signal, and $y[n]$ is the filtered signal. $h[k]$ is a vector of the filter coefficients for the lowpass FIR filter. The length of the vector of filter coefficients is N. L is, again, an interpolation factor. By combining the two relationships in Eq (5.16) and the following,

$$Lm = n\text{-}k \tag{5.17}$$

upsampling and filtering can be written as

$$y[n] = \sum_{m=\lceil n-N+1/L \rceil}^{\lfloor n/L \rfloor} h[n\text{-}Lm]x[m]. \tag{5.18}$$

where $\lceil\ \rceil$ and $\lfloor\ \rfloor$ represent the ceiling and floor functions, respectively. The number of algebraic operations is reduced in this equation using the knowledge that $v[l]$ is zero at many points.

Downsampling by a factor of M is formulated as

$$\begin{aligned} z[j] &= y[jM] \\ &= \sum_{m=\lceil jM-N+1/L \rceil}^{\lfloor jM/L \rfloor} h[jM\text{-}Lm]x[m] \qquad j=0,\ \pm 1, \pm 2, \ldots \end{aligned} \tag{5.19}$$

where $z[j]$ is the downsampled signal. The outputs do not need to be calculated at all of the data points of the upsampled signal, as can be seen in Eq. (5.19). The output needs to be calculated only at every M-th data point of the upsampled signal. If an IIR filter was employed, the filter would need outputs at all of the data points of the upsampled signal. This polyphase implementation, thus, reduces the number of numerical operations involved in resampling. However, implementation is still challenging if the upsampling factor, L, is large.

Resampling with a noninteger downsampling factor

FIR filter design becomes extremely challenging when sampling frequencies need to be precisely converted. For example, resampling of a signal from 100.01 to 150 Hz requires a lowpass filter with a cutoff frequency of $\pi/15000$. The filter needs tens of thousands of filter coefficients. Design of such a filter is computationally challenging. Also a large number of filter coefficients may not fit in the available memory on smart sensors. This problem is addressed mainly by introducing linear interpolation.

First, the resampling is generalized by introducing an initial delay, l_i. Eq. (5.19) is rewritten with l_i as follows:

$$
\begin{aligned}
z[j] &= y[jM + l_i] \\
&= \sum_{m=\lceil (jM + l_i - N + 1)/L \rceil}^{\lfloor (jM+l_i)/L \rfloor} h[jM+l_i\text{-}Lm]x[m] \qquad j=0,\ \pm 1, \pm 2, \ldots
\end{aligned}
\tag{5.20}
$$

By introducing l_i, the beginning point of the downsampled signal can be finely adjusted. If the time difference at the start of sensing is taken into account by l_i, the synchronization accuracy of signals is not limited by the original sampling period.

Resampling is then combined with linear interpolation to achieve the necessary accuracy. The integer, M is replaced by a real number, M_r. The upsampling rate, L_a, must remain an integer. M_r and L_a are not uniquely determined. These values are chosen so that L_a is not too large. A large value of L_a requires a high-order lowpass filter, as is the case for the normal polyphase implementation of resampling. Using these upsampling and downsampling factors, resampling is performed. Upsampling is same as before. However, the downsampling process shown in Eq. (5.20) cannot be directly applied, because of the noninteger downsampling factor. Output data points to be calculated do not necessarily correspond to points on the upsampled signal. Output data points often fall between upsampled data points. Linear interpolation is used to calculate output values as follows:

$$
\begin{aligned}
z[j] &= y[p_l] \cdot (p_u - p_j) + y[p_u] \cdot (p_j - p_l) \\
&= \sum_{m=\lceil (p_l - N + 1)/L_a \rceil}^{\lfloor p_l/L_a \rfloor} h[p_l\text{–}L_a m]x[m] \cdot (p_u - p_j) \\
&\quad + \sum_{m=\lceil (p_u - N + 1)/L_a \rceil}^{\lfloor p_u/L_a \rfloor} h[p_u\text{–}L_a m]x[m] \cdot (p_j - p_l) \\
p_j &= jM_r + l_i \\
p_l &= \lfloor jM_r + l_i \rfloor \\
p_u &= p_l + 1
\end{aligned}
\tag{5.21}
$$

In this way, resampling of an arbitrary noninteger rational factor can be achieved. When L_a is not too small, the approximation of linear interpolation gives reasonable results. A value of L_a ranging from 20 to 150 is employed in algorithmic testing and shown to give reasonable results.

Implementation on Imote2s

The proposed resampling approach is employed to address issues toward synchronized sensing. This approach is first overviewed.

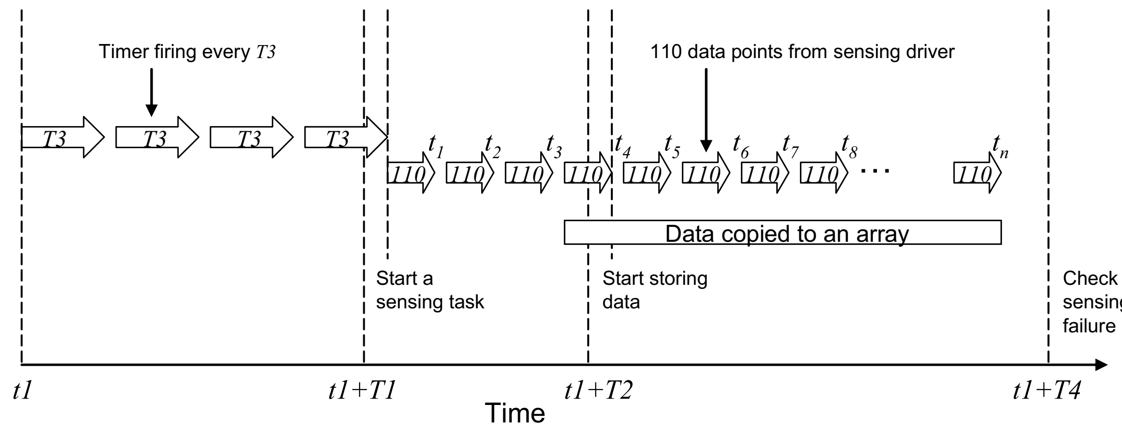


Figure 5.26. Time stamps and waiting times used for synchronized sensing.

The time stamp to indicate the beginning of the entire sensing process, $t1$, and the predetermined waiting times, $T1$, $T2$, and $T3$, are utilized to implement this resampling approach as well as the time stamp at the end of each block of data, t_i $(i=1,2, ..., n)$. $T1$ is the waiting time before the sensing task is posted. A timer component that fires every $T3$ checks whether the waiting time $T1$ has passed. $T2$ is the waiting time before the Imote2 starts storing the measured data. The time stamps, t_i, are utilized later to compensate for misalignment of the starting time, as well as for individual differences in sampling frequency and the time varying sampling frequency. Figure 5.26 illustrates the use of these time stamps and waiting times. This figure also shows the waiting time $T4$ for sensing failure detection. If sensing has not finished at $t1+T4$, the sensing is restarted as previously stated. The details of this approach are explained in the subsequent paragraphs.

When smart sensors are ready to begin sensing, the manager sensor gets a global time stamp, $t1$, and multicasts it to the other nodes. The predetermined waiting times, $T1$ and $T2$, are also multicast. All of the nodes then start a timer component, which is set to be fired every $T3$. When the timer is fired, the global time is checked. If the global time is larger than $t1+T1$, a task to start sensing is posted and the periodic timer stops firing. When a block of sensed data is available, an event is signaled. In this event, the global time is checked again. If the global time is greater than $t1+T2$, then the block of data is stored in memory. Otherwise, the block of data is discarded. From the time stamp of the last data point of the current block, $t_{current}$, that of the last block, t_{last}, and $t1+T2$, the initial misalignment is estimated. When the number of data points in a block is N_{data}, the misalignment in time of the first data point, t_{ini}, is estimated as follows:

$$t_{ini} = t1+T2-t_{last}-\frac{t_{current}-t_{last}}{N_{data}} \tag{5.22}$$

where t_{ini} is later used in resampling as l_i in Eq. (5.21) for the first block of data to compensate for misalignment of the starting time (see Figure 5.27).

Timestamps and resampling also compensate for difference and fluctuation in the sampling frequency (see Figure 5.28). The sampling frequency of the current data block, $fs_{current}$, is estimated from $t_{current}$ and t_{last}.

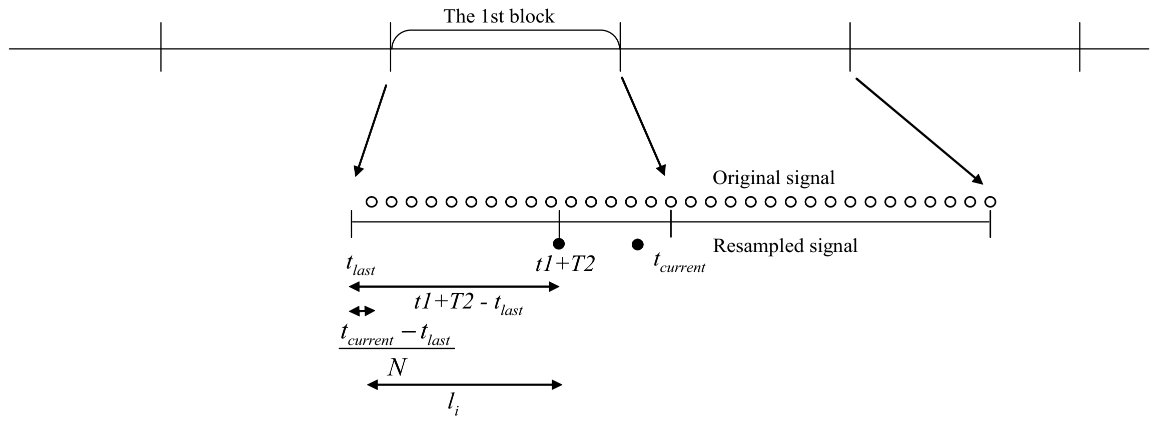


Figure 5.27. Resampling of the first block of data.

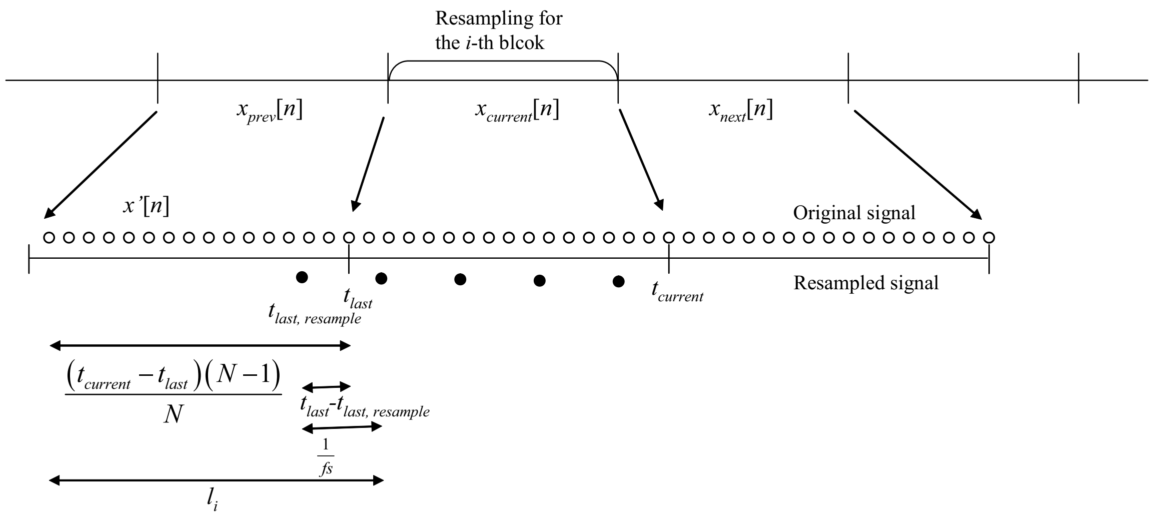


Figure 5.28. Resampling of the *i*-th block of data (*i*>1).

$$fs_{current} = \frac{t_{current} - t_{last}}{N_{data}} \tag{5.23}$$

The sampling rate conversion by a factor L_a / M_r is applied to the block of data. The factor is determined by

$$\begin{aligned} L_a &= fs / d_s \\ M_r &= fs_{current} / d_s \end{aligned} \tag{5.24}$$

where d_s is the common denominator introduced to have the integer, L_a, in an appropriate range. fs is the sampling rate after the rate conversion. l_i in Eq. (5.21) for the first block of data is t_{ini}. For the subsequent blocks, l_i is determined by the time of the last resampled signal of the previous block, $t_{last,resample}$. Because Eq. (5.21) requires $x[m]$ in blocks before or after the current block, data in these blocks are also utilized. To be more specific, when resampling is applied to the current block of data, data from the block before and after the current block are used as part of the input signal for the resampling, $x'[n]$, i.e.,

$$x'[n] = \begin{cases} x_{prev}[n] & 0 \le n < N_{data} \\ x_{current}[n\text{-}N_{data}] & N_{data} \le n < 2N_{data} \\ x_{next}[n\text{-}2N_{data}] & 2N_{data} \le n < 3N_{data} \end{cases} \tag{5.25}$$

where x_{prev}, $x_{current}$, and x_{next} are data in the previous, current, and next blocks, respectively. The sampling frequency of $x'[n]$ is assumed to be same as that of $x_{current}[n]$. l_i for $x'[n]$ is then calculated as

$$l_i = (t_{current} - t_{last}) \cdot (N_{data} - 1)/N_{data} - (t_{last} - t_{last,\ resample}) + \frac{1}{fs} \tag{5.26}$$

Using these parameters, Eq. (5.21) is applied to each data block.

This resampling-based approach, however, cannot be applied on-the-fly by the Imote2s. The resampling is applied after all of the Imote2s have acquired signals.

This approach can also address differences in sampling frequency among nodes and fluctuation of the frequency over time. Due to the denominator, d_s in Eq. (5.24), the upsampling factor is kept moderate, eliminating the need for a lowpass filter with an extremely large number of filter coefficients.

While this algorithm can be implemented on Imote2s to achieve synchronized sensing, numerical operations are nontrivial. Eq. (5.21) still needs a large number of multiplications, additions, etc.; the number of coefficients is still large, frequently greater than a thousand. One thousand filter coefficients, for example, occupy 8 kB of memory space. These issues are addressed by employing integer operations when applicable.

As is apparent from Eq. (5.15), scaling FIR filter coefficients results in filter outputs scaled by the same factor. FIR filter coefficients are multiplied by a large constant, η, so that these coefficients can be well approximated by 16-bit integers.

$$h'[n] = \lfloor h[n] \cdot \eta + 0.5 \rfloor \tag{5.27}$$

These 16-bit integers representing $h'[n]$ are stored on the Imote2s instead of 64-bit double precision data, saving considerable memory. The scaled output, $y[p_l]$ and $y[p_u]$ in Eq. (5.28) are estimated only by integer operations; $h'[n]$ and $x[n]$ are both integer type variables.

$$
\begin{aligned}
z[j] &= (y[p_l]\cdot(p_u-p_j)+y[p_u]\cdot(p_j-p_l))/\eta \\
&= \left(\sum_{m=\lceil (p_l-N+1)/L_a\rceil}^{\lfloor p_l/L_a\rfloor} h'[p_l-L_a m]x[m]\cdot(p_u-p_j)\right. \\
&\quad \left. + \sum_{m=\lceil (p_u-N+1)/L_a\rceil}^{\lfloor p_u/L_a\rfloor} h'[p_u-L_a m]x[m]\cdot(p_j-p_l)\right)/\eta \qquad (5.28)\\
p_j &= jM_r+l_i \\
p_l &= \lfloor jM_r+l_i\rfloor \\
p_u &= p_l+1
\end{aligned}
$$

Then linear interpolation is performed by casting all associated integers to double precision data. The final outcome is adjusted to account for the scaling factor of the filter coefficients. In this way, implementation of the proposed resampling approach becomes less numerically challenging.

5.4 Summary

Middleware services for smart sensors were developed in this chapter. Model-based data aggregation, including distribution and coordination, provided scalability to a large number of smart sensors without sacrificing performance of the SHM algorithms. The data loss problem, which was shown to have adverse effects on an SHM algorithm, was addressed by developing reliable communication protocols. To realize synchronized sensing, a resampling-based approach was proposed. These middleware services are implemented on the Imote2 running TinyOS. In Chapter 7, these middleware services will be used in realization of an SHM system.

Chapter 6

ALGORITHMS

In this chapter, algorithms for SHM to be implemented on smart sensors are discussed. The Distributed Computing Strategy (DCS) for SHM proposed by Gao (2005) has the potential to realize densely deployed networks of smart sensors for SHM, because of its local data sharing and processing. The algorithmic components of DCS for SHM, i.e., Natural Excitation Technique (NExT), Eigensystem Realization Algorithm (ERA), the Damage Locating Vector (DLV) method, and DCS, are briefly reviewed. Though most of the data processing is performed locally, the initialization phase of the strategy to estimate mode shape normalization constants involves more cumbersome processes; the initialization requires either input force measurement or output measurement before and after a known mass perturbation. Recent algorithmic developments of a stochastic DLV (SDLV) method by Bernal (2006) allows estimation of DLVs without input force measurement or mass perturbation. DCS is extended with this stochastic DLV method to allow for less demanding initialization.

6.1 Natural Excitation Technique

To understand the NExT (James et al., 1992, 1993) considered the equation of motion in Eq. (6.1) under the assumption of the stationary random process.

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{f}(t) \tag{6.1}$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are the n x n mass, damping, and stiffness matrices, respectively; $\mathbf{x}(t)$ is a n x 1 displacement vector; $\mathbf{f}(t)$ is a m x 1 force vector; $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$ are the velocity and the acceleration vectors, respectively. By multiplying the displacement at the reference sensor and taking the expected value, Eq. (6.1) is transformed as follows:

$$\begin{aligned}&\mathbf{M}\mathrm{E}[\ddot{\mathbf{x}}(t+\tau)x_{\mathrm{ref}}(t)] + \mathbf{C}\mathrm{E}[\dot{\mathbf{x}}(t+\tau)x_{\mathrm{ref}}(t)] + \mathbf{K}\mathrm{E}[\mathbf{x}(t+\tau)x_{\mathrm{ref}}(t+\tau)] \\ &\quad = \mathrm{E}[\mathbf{f}(t+\tau)x_{\mathrm{ref}}(t)]\end{aligned} \tag{6.2}$$

Because the input force and response at the reference sensor location are uncorrelated for $\tau \geq 0$, the right-hand side of Eq. (6.2) is zero. The expectation between the two signals is the correlation function. Therefore, by denoting $\mathrm{E}[\mathbf{x}(t+\tau)\mathbf{y}(t)]$ as the correlation function $\mathbf{R}_{\mathbf{xy}}(\tau)$, Eq. (6.2) is rewritten as

$$\mathbf{M}\mathbf{R}_{\ddot{\mathbf{x}}x_{\mathrm{ref}}}(\tau) + \mathbf{C}\mathbf{R}_{\dot{\mathbf{x}}x_{\mathrm{ref}}}(\tau) + \mathbf{K}\mathbf{R}_{\mathbf{x}x_{\mathrm{ref}}}(\tau) = 0, \tau \geq 0 \tag{6.3}$$

When $A(t)$ and $B(t)$ are weakly stationary processes, the following relation holds (Bendat & Piersol, 2000).

$$\mathbf{R}_{\dot{A}B}(\tau) = \dot{\mathbf{R}}_{AB}(\tau) \tag{6.4}$$

A similar relation holds for higher derivatives,

$$\mathbf{R}_{A^{(m)}B}(\tau) = \mathbf{R}^{(m)}{}_{AB}(\tau) \tag{6.5}$$

where the superscript, m, denotes the m-th derivative. Consequently, Eq.(6.3) can be rewritten as

$$\mathbf{M}\ddot{\mathbf{R}}_{\mathbf{x}x_{\mathrm{ref}}}(\tau) + \mathbf{C}\dot{\mathbf{R}}_{\mathbf{x}x_{\mathrm{ref}}}(\tau) + \mathbf{K}\mathbf{R}_{\mathbf{x}x_{\mathrm{ref}}}(\tau) = 0, \tau \geq 0 \tag{6.6}$$

Thus, correlation functions for the stationary responses are shown to satisfy the equation of motion for free vibration. This fact can be directly used for the subsequent modal analysis.

6.2 Eigensystem Realization Algorithm

ERA (Juang & Pappa, 1985) identifies modal parameters from free vibration responses. When measurement at p sensors are available in a measurement vector $\mathbf{y}[n]$, the Markov parameters $\mathbf{Y}[n]$ from m sets of measurement are constructed as follows:

$$\mathbf{Y}[n] = \left[\mathbf{y}[1], \mathbf{y}[2], \ldots, \mathbf{y}[m]\right] \tag{6.7}$$

A generalized Hankel matrix, $\mathbf{H}_{rs}[k-1]$, is formed as a $r \times s$ block matrix:

$$\mathbf{H}_{rs}[k-1] = \begin{bmatrix} \mathbf{Y}[k] & \mathbf{Y}[k+j_1] & \ldots & \mathbf{Y}[k+j_{s-1}] \\ \mathbf{Y}[h_1+k] & \mathbf{Y}[h_1+k+j_1] & \ldots & \mathbf{Y}[h_1+k+j_{s-1}] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}[h_{r-1}+k] & \mathbf{Y}[h_{r-1}+k+j_1] & \ldots & \mathbf{Y}[h_{r-1}+k+j_{s-1}] \end{bmatrix} \tag{6.8}$$

A SVD of this Hankel matrix yields

$$\mathbf{H}_{rs}[0] = \mathbf{PDQ}^{\mathrm{T}} \tag{6.9}$$

where the superscript T denotes the matrix transpose. Components in these matrices corresponding to small singular values are considered noise and replaced by zeroes. Juang and Pappa (1985) derived that the triple, $\{\mathbf{D}^{-1/2}\mathbf{P}^{\mathrm{T}}\mathbf{H}_{rs}[1]\mathbf{QD}^{-1/2}, \mathbf{D}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{E}_m, \mathbf{E}_p^{\mathrm{T}}\mathbf{PD}^{1/2}\}$ is a minimum realization of the measured system. A matrix to extract the first m columns, $\mathbf{E}_m$ is defined using an identity matrix of order m and a null matrix of size m.

$$\mathbf{E}_m^{\mathrm{T}} = [\mathbf{I}_m, 0_m, 0_m, \ldots, 0_m] \tag{6.10}$$

The eigenproblem of the system matrix is then solved:

$$(\mathbf{D}^{-1/2}\mathbf{P}^{\mathrm{T}}\mathbf{H}_{rs}[k]\mathbf{Q}\mathbf{D}^{-1/2})\psi_i = z_i\psi_i \tag{6.11}$$

The natural frequencies f_i and damping ratios ζ_i are then calculated from the eigenvalues, z_i as

$$\begin{aligned}\lambda_i &= \ln z_i/(k\Delta t)\\ f_i &= |\lambda_i|/(2\pi)\\ \zeta_i &= \mathrm{Re}(\lambda_i)/|\lambda_i|\end{aligned} \tag{6.12}$$

where $\ln z_i$ is the principal value of natural logarithm of z_i. Δt is the sampling period. $\mathrm{Re}(\ )$ takes the real part of a complex number. The mode shape, ϕ_i, corresponding to z_i is calculated as

$$\phi_i = \mathbf{E}_p^{\mathrm{T}}\mathbf{P}\mathbf{D}^{1/2}\psi_i. \tag{6.13}$$

The initial modal amplitudes, Θ, which can be utilized to calculate modal amplitude coherence, an indicator to quantitatively distinguish the system and noise modes, is estimated as follows:

$$\Theta = \psi^{-1}\mathbf{D}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{E}_m \tag{6.14}$$

6.3 Damage Locating Vector method

One of the flexibility-based SHM approaches is the Damage Locating Vector (DLV) method, which is briefly described here. The DLV method, first developed by Bernal (2002), is based on the determination of a special set of vectors, the so-called damage locating vectors (DLVs). These DLVs have the property that when they are applied to the structure as static forces at the sensor locations, no stress is induced in the damaged elements. This unique characteristic can be employed to localize structural damage (Gao, 2005).

For a linear structure, the flexibility matrix, $\mathbf{F}$, at the sensor locations is constructed from measured data. Gao (2005) explained two approaches to estimate the flexibility matrix. Both of them reconstruct the flexibility matrix using mode shapes and normalization constants. One approach estimates the normalization constants assuming that the input force can be measured, while the other utilizes output-only measurements before and after a known mass perturbation.

Formulation of the flexibility matrix based on input force measurements is summarized by the following equation involving complex conjugate pairs of arbitrary normalized mode shapes, $\Psi = \lfloor \varphi\ \varphi^* \rfloor$.

$$\mathbf{F} = -\Psi \mathbf{D_g} \Psi^{\mathrm{T}} \tag{6.15}$$

$\mathbf{D_g} = diag\left(\left[d_1\ d_2\ \dots\ d_j\ \dots\right]\right)$ is the matrix of modal normalization constant, d_j. If all of the modes are observed and available for reconstruction, $\mathbf{F}$ can be estimated. When only a part of structural modes is available, Eq. (6.19) gives an approximation of $\mathbf{F}$. Gao (2005) showed that $\mathbf{F}$ can be approximated well from a limited number of modes.

The normalization constants can be estimated through the following relation:

$$d_j = \lambda_j^{-(p+1)} \bar{\phi}_j^{\mathrm{T}} \mathbf{B} \mathbf{q_f} [diag(\Psi_{m,j}^{\mathrm{T}} \mathbf{q}_m)]^{-1} \tag{6.16}$$

where mode shapes at sensor locations Ψ_m and eigenvectors ϕ satisfies the following relationship:

$$\Psi_m = \mathbf{C}\phi \tag{6.17}$$

$$\mathbf{A}\phi = \phi\Lambda \tag{6.18}$$

$\bar{\phi}_j^{\mathrm{T}}$ is the j-th row of the inverse of eigenvector matrix ϕ^{-1}, and $\Psi_{m,j}^{\mathrm{T}}$ is the j-th row of the transposed mode shape matrix Ψ_m^{T}. Λ is a diagonal matrix of eigenvalues. When there is more than one colocated sensor and actuator pair, multiple estimates of d_j will be obtained. Bernal and Gunes (2004) suggested that d_j corresponding to the component in vector $\Psi_{m,j}^{\mathrm{T}} \mathbf{q}_m$ with the largest magnitude might be used. Using the normalization constants, the flexibility matrix is estimated as in Eq. (6.15); only a part of the flexibility matrix corresponding to sensor locations is practically estimated by replacing the mode shape matrix Ψ with the matrix of mode shapes at sensor location, Ψ_m.

When the measurement of input force is impractical, output-only measurements before and after a known mass perturbation are utilized to estimate normalization constants for flexibility matrix reconstruction. The estimated mode shapes and eigenvalues, as well as normalization constants, reconstruct $\mathbf{F}$ as follows:

$$\begin{aligned} \mathbf{F} &= \Phi \Lambda^{-1} \Phi^{\mathrm{T}} \\ &= (\phi\alpha)\Lambda^{-1}(\phi\alpha)^{\mathrm{T}} \end{aligned} \tag{6.19}$$

where Φ is the matrix of mass normalized mode shapes, and ϕ is the matrix of arbitrarily normalized mode shapes. α is a diagonal matrix of mass normalization constants, α_i. Note that estimation of normalization constants from a known mass perturbation is theoretically derived for a proportionally damped system; mode shapes should be real. An approximate flexibility matrix is estimated from a limited number of modes.

In the mass perturbation approach (Bernal, 2004), the mode shapes and eigenvalues are determined before and after a known mass perturbation is introduced to the structure. The mass matrix of the modified structure, $\mathbf{M}_1$, is expressed as

$$\mathbf{M}_1 = \mathbf{M}_0 + \Delta\mathbf{M} \tag{6.20}$$

where $\mathbf{M}_0$ is the mass matrix of the original structure, and $\Delta\mathbf{M}$ represents the mass perturbation. α_i is estimated from modal parameters and $\Delta\mathbf{M}$ as follows:

$$\alpha_i^2 = \frac{\lambda_{0,i} - \lambda_{1,i}}{\lambda_{1,i}} \cdot \frac{q_{ii}}{\phi_{0,i}^{\mathrm{T}} \Delta\mathbf{M} \phi_{1,i}^{\mathrm{T}}} \tag{6.21}$$

$$\mathbf{q}_i = \begin{bmatrix} q_{1i} \; q_{2i} \; \dots \; q_{ni} \end{bmatrix}^{\mathrm{T}} = (\phi_0^{\mathrm{T}} \phi_0)^{-1} \phi_0^{\mathrm{T}} \phi_{1,i} \tag{6.22}$$

The subscripts 0 and 1 indicate quantities before and after the mass perturbation, respectively, and n is the number of observed modes. While Eq. (6.22) uses mode shapes at all DOFs, the usage of mode shapes at sensor location is shown to give reasonable results (Gao, 2005). ϕ in Eq. (6.19) is in practice replaced with mode shapes at sensor locations, yielding a corresponding flexibility matrix.

Estimation of the flexibility matrix using either input force measurement or known mass perturbation is performed before and after damage to localize the damage. Estimated flexibility matrices of the undamaged and damaged structures are denoted as $\mathbf{F_u}$ and $\mathbf{F_d}$, respectively. All of the linearly independent load vectors $\mathbf{L}$ are collected, which satisfy the following relationship:

$$\mathbf{F_d L} = \mathbf{F_u L} \text{ or } \mathbf{F}_\Delta \mathbf{L} = (\mathbf{F_d} - \mathbf{F_u})\mathbf{L} = 0 \tag{6.23}$$

$\mathbf{F}_\Delta$ is change in the flexibility matrix. This equation implies that the load vectors $\mathbf{L}$ produce the same displacements at the sensor locations before and after damage. From the definition of the DLVs, these vectors also satisfy Eq. (6.23); that is, because the DLVs induce no stress in the damaged elements, damage in those elements does not affect the displacements at the sensor locations. Therefore, DLVs are indeed the vectors in $\mathbf{L}$. To calculate $\mathbf{L}$, SVD is employed. The SVD of the matrix $\mathbf{F}_\Delta$ leads to

$$\mathbf{F}_\Delta = \mathbf{USV}^{\mathrm{T}} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_0 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^{\mathrm{T}} \\ \mathbf{V}_0^{\mathrm{T}} \end{bmatrix} \tag{6.24}$$

Recall from the characteristics of SVD that

$$\begin{bmatrix} \mathbf{V}_1^{\mathrm{T}} \\ \mathbf{V}_0^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_0 \end{bmatrix} = \mathbf{I} \tag{6.25}$$

where $\mathbf{I}$ is identity matrix. Eq.(6.24) can then be rewritten as

$$\begin{bmatrix} \mathbf{F}_\Delta \mathbf{V}_1 & \mathbf{F}_\Delta \mathbf{V}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{U}_1 \mathbf{S}_1 & 0 \end{bmatrix} \tag{6.26}$$

From Eq. (6.26), one obtains

$$\mathbf{F}_\Delta \mathbf{V}_0 = 0 \tag{6.27}$$

Eqs. (6.23) and (6.27) indicate that $\mathbf{L} = \mathbf{V}_0$, i.e., DLVs can be obtained from the SVD of the difference matrix $\mathbf{F}_\Delta$. In Eq. (6.24), because of noise and computational errors, the singular values corresponding to $\mathbf{V}_0$ are not exactly zero. To select DLVs from SVD results of the matrix $\mathbf{F}_\Delta$, Bernal (2002) proposed an index *svn* defined as

$$svn_i = \sqrt{\frac{s_i c_i^2}{\max\limits_k (s_k c_k^2)}} \tag{6.28}$$

where s_i is the *i*-th singular value of the matrix $\mathbf{F}_\Delta$; c_i is the constant that is used to normalize the maximum stress in the structural element, which is induced by the state load $c_i \mathbf{V}_i$, to have a value of one; and $\mathbf{V}_i$ is the right singular vector of $\mathbf{F}_\Delta$.

Each of the DLVs is then applied to an undamaged analytical model of the structure, and the stress in each structural element is calculated. If an element has zero normalized accumulative stress, this element is a candidate damaged element. The normalized accumulative stress for the *j*-th element is defined as

$$\bar{\sigma}_j = \frac{\sigma_j}{\max\limits_k (\sigma_k)} \tag{6.29}$$

where

$$\sigma_j = abs\left(\sum_{i=1}^{n} \frac{\sigma_{ij}}{\max\limits_k (\sigma^i_k)}\right) \tag{6.30}$$

In Eq. (6.30), σ_{ij} is the stress in the *j*-th element induced by the *i*-th DLV; σ_j is the cumulative stress in the *j*-th element. In practice, the normalized accumulative stress induced by DLVs in the damaged elements may not be exactly zero due to noise and uncertainties. A reasonable threshold should be chosen to select damaged elements.

6.4 Distributed Computing Strategy for SHM

Gao (2005) proposed the Distributed Computing Strategy (DCS) for SHM employing the algorithms reviewed in previous sections of this chapter. In this approach, a hierarchy of local sensor communities is organized, so that the data does not need to be centrally collected or analyzed. Because these algorithms are applicable to data only from nodes in a local sensor community, the analysis can be performed within a local sensor community independent of the other nodes; data from a given community is processed within the same community, eliminating the need to globally transfer a large amount of data through a long communication path.

While most of the derivation in NExT, ERA, and the DLV method can be directly applicable to data from a local sensor community, the mass normalization constants, d_j or α_i, for the DLV method may need data at DOFs outside the community. To estimate d_j in a local sensor community, the input force and the output response at that location need to be available. α_i needs mode shapes to be estimated for DOFs corresponding to the location of the mass perturbation, as shown in Eq. (6.21). In addition, q_{ii} in Eq. (6.22) may contain a large error if only a limited number of sensors in a local area is used. Gao (2005) suggested that estimation of mass normalization constants be performed from globally obtained data during initialization of the SHM system. Normalization constants obtained during initialization are then converted to normalization constants for each sensor community through the following equations:

$$d_j^i = d_j \cdot \left(\frac{\Psi_{j,\,\mathrm{ref}(i)}}{\Psi_{j,\,\mathrm{ref}}^i} \right)^2 \tag{6.31}$$

$$\alpha_j^i = \alpha_j \cdot \frac{\phi_{j,\,\mathrm{ref}(i)}}{\phi_{j,\,\mathrm{ref}}^i} \tag{6.32}$$

d_j^i and α_j^i are the j-th mode mass normalization constants for the sensor community, i. $\Psi_{j,\,\mathrm{ref}(i)}$ and $\phi_{j,\,\mathrm{ref}(i)}$ are the j-th global mode shape's components corresponding to the reference sensor node of the i-th sensor community. $\Psi_{j,\,\mathrm{ref}}^i$ and $\phi_{j,\,\mathrm{ref}}^i$ are components of j-th mode shape determined in the i-th sensor community. These components correspond to the reference sensor DOF of the community. In this way, mass normalization constants for each sensor community can be estimated. Thus, monitoring is realized within local sensor communities.

DCS also describes how local sensor communities collaborate with each other to judge structural damage, including redundancy to increase fault tolerance. Local sensor communities are formed with overlap, so that more than one sensor community monitors the same element. This overlap provides the means to deal with false-positive and false-negative damage detection. Information about localized damaged elements is exchanged among local sensor communities. If damage localization in one sensor community contradicts that in another community, measurement and analysis are repeated. If a damaged element is detected and no inconsistency is reported, the base station is informed of the damage (see Figure 6.1).

6.5 Stochastic Damage Locating Vector method

Recent algorithmic development of the Stochastic DLV (SDLV) method by Bernal (2006) has the potential to improve the DCS for SHM by eliminating the need to estimate normalization constants. The initialization phase of DCS involving either input force

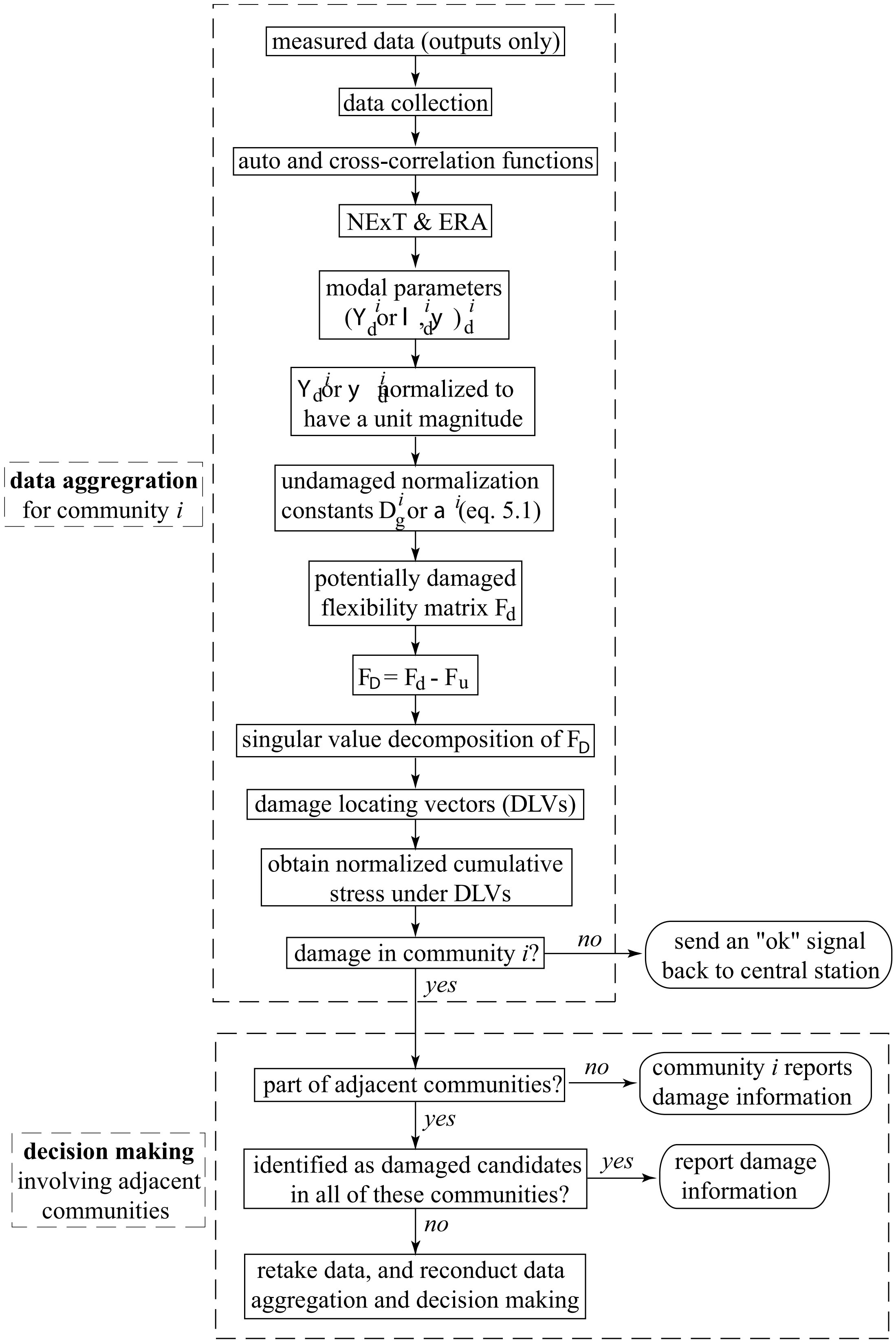

measured data (outputs only)
data collection
auto and cross-correlation functions
NExT & ERA
modal parameters
$(\Psi_d^i$ or $\lambda_d^i, \psi_d^i)$
Ψ_d^i or ψ_d^i normalized to have a unit magnitude
undamaged normalization constants D_g^i or α^i (eq. 5.1)
potentially damaged flexibility matrix F_d
$F_D = F_d - F_u$
singular value decomposition of F_D
damage locating vectors (DLVs)
obtain normalized cumulative stress under DLVs
damage in community *i*?
no
send an "ok" signal back to central station
yes
data aggregration for community *i*
part of adjacent communities?
no
community *i* reports damage information
yes
identified as damaged candidates in all of these communities?
yes
report damage information
no
retake data, and reconduct data aggregation and decision making
decision making involving adjacent communities

Figure 6.1. DCS logic (Gao, 2005).

measurement or mass perturbation can be greatly simplified with the SDLV method. The SDLV method is briefly reviewed in this section.

Bernal (2006) stated that the null space of the change in the flexibility matrix will be contained in the null of $\Delta\mathbf{Q}^{\mathrm{T}}$. The matrix $\Delta\mathbf{Q}^{\mathrm{T}}$ is the transpose of the change in $\mathbf{Q}$ which is defined as follows:

$$\mathbf{Q} = -\mathbf{C}\mathbf{A}^{-(p+1)}\mathbf{H}_p^{\dagger}\mathbf{L} \tag{6.33}$$

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{C}\mathbf{A}^{1-p} \\ \mathbf{C}\mathbf{A}^{-p} \end{bmatrix} \tag{6.34}$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} \tag{6.35}$$

where $\mathbf{I}$ denotes identity matrix, 0 denotes a zero matrix. $p = 0$, 1, or 2, depending on whether the measured outputs are displacement, velocity, or acceleration. The system matrix $\mathbf{A}$ and observation matrix $\mathbf{C}$ are determined through modal analysis such as ERA. The null vectors of $\Delta\mathbf{Q}^{\mathrm{T}}$ are treated as DLVs.

Bernal (2006) also proposed a way to determine the number of DLVs and to combine information from multiple DLVs. When there are q DLVs and the stress corresponding to each vector is σ_j, a normalized stress index (*nsi*) is defined as follows:

$$nsi_j = \frac{|\sigma_j|}{|\sigma_j|_{max}} \tag{6.36}$$

Weighting can be incorporated to yield a weighted stress index (*WSI*) as

$$WSI = \sum_{j=1}^{q} \gamma_j nsi_j \tag{6.37}$$

where γ_j is a weighting parameter. Potentially damaged elements (PD) are chosen as

$$PD = \{elements | WSI \le tol\} \tag{6.38}$$

Bernal (2006) took $\gamma_j = 1$ and $tol = 0.1 \cdot (WSI)_{max}$. The number of DLVs, q, can be estimated as

$$q = 0.5 \cdot \left\{ \#\, of\, \gamma\ values \le 0.1 \middle| \gamma = \sqrt{\frac{s_i}{s_j|_{max}}} \right\} \tag{6.39}$$

where s_j are the singular values of $\Delta\mathbf{Q}$. Bernal (2006) demonstrated that the proposed *WSI* successfully indicates damage in a numerical truss structure.

6.6 Extension of DCS for SHM with the SDLV method

The SDLV method simplifies DCS for SHM. The SDLV method does not require normalization constant estimation, eliminating the need for initialization with input force measurement or mass perturbation. The derivation does not assume response measurement at locations distributed over the entire structures. Structural responses are measured only within local sensor communities before and after damage, and the SDLV method is applied subsequently.

There are several ways to construct the system matrix $\mathbf{A}$ and the observation matrix $\mathbf{C}$. The triple $\{\mathbf{D}^{-1/2}\mathbf{P}^{\mathrm{T}}\mathbf{H}_{rs}[1]\mathbf{Q}\mathbf{D}^{-1/2}, \mathbf{D}^{1/2}\mathbf{Q}^{\mathrm{T}}\mathbf{E}_m, \mathbf{E}_p^{\mathrm{T}}\mathbf{P}\mathbf{D}^{1/2}\}$, determined in ERA is a minimum realization of the system; the first and third terms can be used as $\mathbf{A}$ and $\mathbf{C}$. However, in practice, the triple may contain noise modes. Another way to formulate $\mathbf{A}$ and $\mathbf{C}$ matrices is to use modal parameters, i.e.,

$$\mathbf{A} = diag\left(\left[\lambda_1\ \lambda_2\ \ldots\ \lambda_n\ \lambda_1{*}\ \lambda_2{*}\ \ldots\ \lambda_n{*}\right]\right) \tag{6.40}$$

$$\mathbf{C} = \left[\Psi_1\ \Psi_2\ \ldots\ \Psi_n\ \Psi_1{*}\ \Psi_2{*}\ \ldots\ \Psi_n{*}\right] \tag{6.41}$$

By selecting modes to be used in Eqs. (6.40) and (6.41), only specified modes contribute to the matrices. $\mathbf{C}$ can also be formulated as

$$\mathbf{C} = \left[\Psi_1\ \Psi_2\ \ldots\ \Psi_n\ 0\ 0\ \ldots\ 0\right] \tag{6.42}$$

assuming the observed variables are the real part of the outputs. Following this formulation, $\Delta\mathbf{Q}^{\mathrm{T}}$ becomes a complex matrix yielding complex DLVs and stress field. The absolute value of the complex stress induced by complex DLVs is used as $|\sigma_j|$. Numerical simulation showed the $\mathbf{C}$ matrix formulation in Eq. (6.42) gives more accurate damage detection results than that in Eq. (6.41). One explanation is that the imaginary parts of the $\mathbf{C}$ matrix in Eq. (6.41) are cancelled out in Eq. (6.33), losing certain information; the $\mathbf{C}$ matrix from the other formation maintains the information from the imaginary part. In the subsequent chapters, the formulation in Eq. (6.42) is utilized.

6.7 Summary

Algorithms to be implemented on smart sensors are briefly reviewed in this section. DCS for SHM is proposed as an algorithm enabling distributed SHM employing densely deployed smart sensors. DCS is extended with the SDLV method to eliminate the need for input force measurement or mass perturbation during the initialization. In the next chapter, these algorithms are implemented on the Imote2 platform using the middleware services developed in the previous chapter.

Chapter 7

REALIZATION OF DCS FOR SHM

The Distributed Computing Strategy (DCS) for SHM reviewed in Chapter 6 is developed for the Imote2 sensor platform in this chapter. First, a number of requisite functions, such as FFT and SVD, are developed and their accuracy limitations are evaluated. DCS is then implemented and validated on a component by component basis.

7.1 Functions

DCS involves numerical operations that are not provided by a standard math library. NExT estimates correlation functions using FFT and the inverse FFT. ERA applies SVD to a matrix of real numbers, as in Eq. (6.9). Eq. (6.11) needs a complex eigensolver. If the initial modal amplitudes need to be estimated to distinguish system and noise modes, the inverse of a complex matrix is essential, as shown in Eq. (6.12). Sorting is also needed in ERA, as modes are usually sorted by their natural frequencies. DLV methods involve SVD and sorting. Furthermore, the SDLV method may require SVD on a complex matrix, depending on formulation of an observation matrix, $\mathbf{C}$. These functions need to be implemented on the Imote2 to realize structural health monitoring applications.

Numerical Recipes in C (Press et al., 1992) and the *CLAPACK User's Guide* (Anderson et al., 1999) explain a variety of numerical functions. Necessary functions are coded based on these references. Because the code and libraries need to be cross-compiled for the Xscale processor on the Imote2, instead of the x86 processor on a PC, the codes in *Numerical Recipes in C* or in CLAPACK cannot be directly used. Relevant files are extracted and modified for use with Imote2 applications.

7.1.1 Fast Fourier Transform

Fast Fourier Transform (FFT) is essential for implementation of the NExT algorithms. NExT uses the auto- and cross-correlation functions for the measured systems as input. First, time histories are converted to the frequency domain using the FFT and averaged to produce the power- and cross-spectral densities. The auto- and cross-correlation functions are then obtained by applying the inverse FFT. The Cooley and Turkey (1965) algorithm, using the Danielson and Lanczos Lemma (Danielson & Lanczos, 1942), enables FFT in $O(N\log_2 N)$ operations. Double-precision FFT available as a part of *Numerical Recipes in C* (Press et al., 1992), is modified to be used on the Imote2.

The numerical accuracy of the FFT implementation on Imote2 is examined by comparing FFT results from the Imote2 with those calculated by MATLAB on a PC. A band-limited white noise is generated as a signal to be analyzed, and the FFT is applied to this signal. As seen in Figure 7.1, the FFT implementation on the Imote2 and on the PC

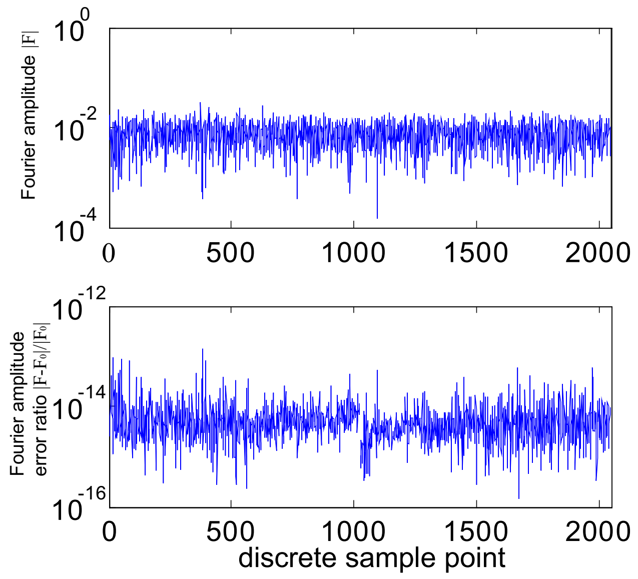


Figure 7.1. FFT accuracy check (Number of FFT data points: 4,096).

give numerically identical FFT results, considering that double precision data has about 15 effective significant digits.

The FFT on double precision data needs $8n$ B memory space to store the data to be analyzed; the data is then replaced with the FFT result. The Imote2's 256 kB of on-board memory can hold about 32,000 double data points. If FFT is applied to a longer record, larger RAM or virtual memory is necessary. Note that the Imote2 has 32 MB of RAM, which can hold a large amount of data. At the time of this research, however, only 256 kB of memory is accessible.

7.1.2 Singular value decomposition

Singular Value Decomposition (SVD) is a part of both the ERA and DLV methods. While SVD in the ERA and mass perturbation DLV method are applied to real valued matrices, the SDLV method may involve SVD of a complex matrix, depending on how the observation matrix, C, is reconstructed through modal analysis. A double precision SVD function for real matrices, as well as one for complex matrices needs to be available.

The source code for SVD of a real matrix is found in *Numerical Recipes in C* and CLAPACK, while CLAPACK also provides source codes for SVD of a complex matrix. First, the matrix is transformed to a bi-diagonal form using the Householder reduction. The bidiagonal matrix is then diagonalized to obtain the singular values. The source code from *Numerical Recipes in C* and CLAPACK are modified for implementation on the Imote2.

The accuracy of the implementation is examined. The singular values of the matrix are chosen as accuracy indicators; singular values of various magnitudes are inspected. An arbitrary matrix is first constructed and decomposed by SVD on MATLAB. The singular values are then replaced with a decreasing geometric series; singular values of this matrix

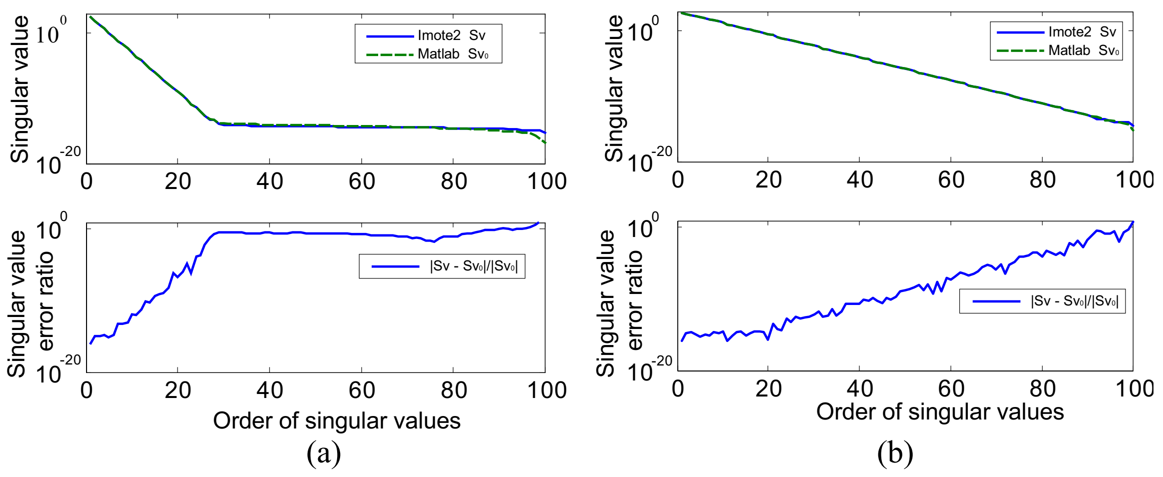


Figure 7.2. SVD accuracy check on 100 x 100 matrix (Numerical recipes functions): (a) smaller singular values; and (b) larger singular values.

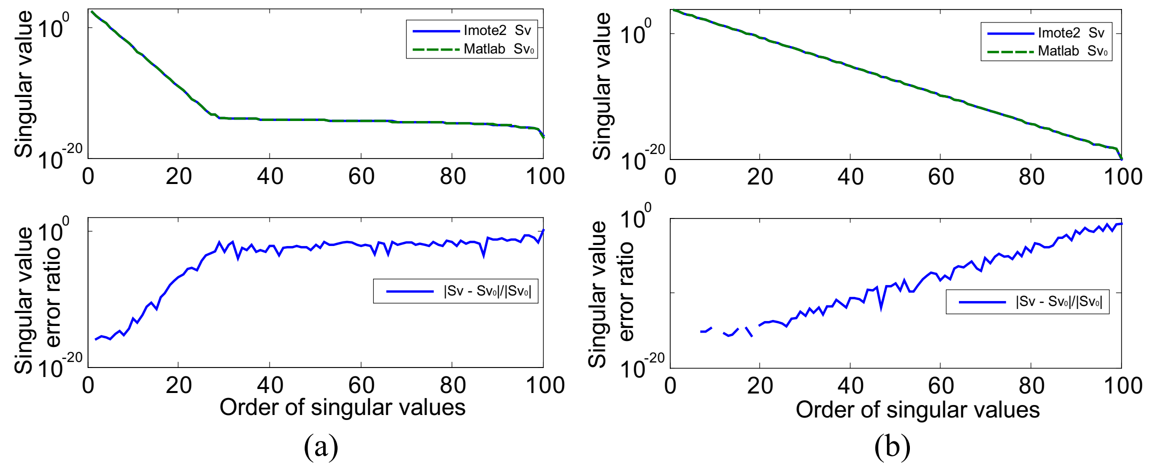


Figure 7.3. SVD accuracy check on 100 x 100 matrix (CLAPACK functions): (a) smaller singular values; and (b) larger singular values.

are consequently distributed equally in log scale, allowing a check of the algorithms for singular values of various magnitudes. A matrix is reconstructed from the calculated singular values and the two unitary matrices obtained in the SVD. This SVD and reconstruction are performed on both the Imote2 and a PC. Figures 7.2 and 7.3 show singular values estimated on the two platforms and the ratio of the numerical difference between the two to the singular values on a PC. The accuracy of the SVD on the Imote2 and on a PC is numerically the same. As another accuracy indicator, the 2-norm of the difference between the original matrix and the matrix reconstructed from the SVD results are calculated as shown in Table 7.1. These values are as small as 10^{-15}. The SVD implementation on the Imote2 is considered to be accurate to within the precision of the data type.

Limitations on the matrix size are also examined. Because smart sensors have limited memory, numerical operations on large matrices are not feasible. An $n \times n$ double precision matrix occupies $8n^2$ B of memory. Therefore the two unitary matrices of SVD

Table 7.1. *SVD Accuracy Check on Matrices of Various Size*

matrix size	$\lvert USV\text{-}A\rvert_2/\lvert A\rvert_2$	
	Numerical Recipes	*CLAPACK*
3x3	1.01 x e-15	1.01 x e-15
30x30	6.39 x e-16	4.96 x e-15
50x50	1.09 x e-15	4.0337 x e-16
100x100	9.72 x e -16	4.1957 x e-15

and the diagonal matrix of singular values require $16n^2 + 8n$ B of memory. Even when the entire 256 kB of RAM on the Imote2 is used for storing these matrices, the maximum size of the matrix to be analyzed is 125×125.

7.1.3 Eigensolver

ERA requires an eigensolver to estimate modal parameters. The eigensolver is employed to the estimated system matrix, **A**. Modal frequencies and damping ratios are estimated from its eigenvalues, while mode shape estimation requires the eigenvectors. Therefore, both eigenvalues and eigenvectors need to be calculated. Moreover, the **A** matrix is not necessarily symmetric. A complex eigensolver capable of calculating both eigenvalues and eigenvectors is needed.

CLAPACK contains a double precision complex eigensolver. The algorithm reduces a matrix to its upper Hessenberg form. The Hessenberg matrix is then reduced to its Schur form. The eigenvalues are obtained as the diagonal elements of the Schur form matrix. The eigenvectors of this matrix is then calculated and transformed back considering the Hessenberg and Schur transforms. The source code is modified for use on the Imote2.

The accuracy of the implementation is examined. A matrix is constructed in the same way as in the SVD function accuracy test. The eigensolver is applied to this matrix on both the Imote2 and the PC. Figure 7.4 shows eigenvalues estimated on the two platforms and the normalized difference between the two eigenvalues. Eigenvalues on the Imote2 and on the PC are considered to be numerically equal. As another accuracy indicator, the Modal Assurance Criterion (MAC) for eigenvectors calculated on the Imote2 and on the PC is investigated. The MAC between two vectors ϕ_1 and ϕ_2 are defined as follows:

$$MAC = \lvert\phi_1{}^* \cdot \phi_2\rvert^2 / (\lvert\phi_1{}^* \cdot \phi_1\rvert \lvert\phi_2{}^* \cdot \phi_2\rvert) \tag{7.1}$$

where * denotes complex conjugate. Identical vectors give a MAC value of 1.0, even if one of them is multiplied by a constant. Uncorrelated vectors give a MAC value of zero. The deviation of the MAC values from one is plotted on Figure 7.5. These values are as small as 10^{-15} when the corresponding eigenvalues are sufficiently large. Although the eigenvectors corresponding to eigenvalues smaller than the precision of the double data type are different from those on the PC, the eigensolver implementation on Imote2 is considered to be accurate with the precision of the data type.

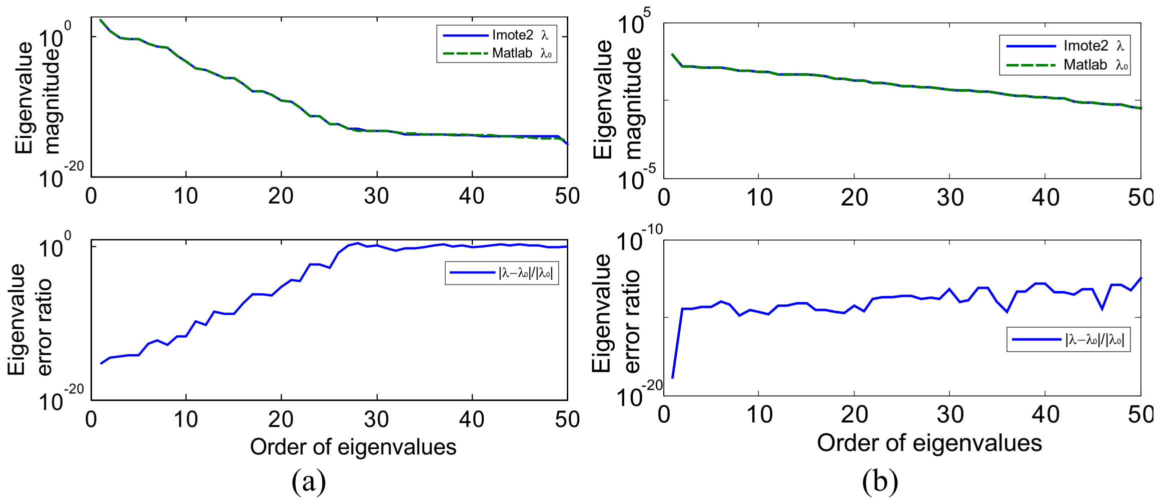


Figure 7.4. Complex eigensolver accuracy check on 50 x 50 matrix (eigenvalues): (a) smaller eigenvalues; and (b) larger eigenvalues.

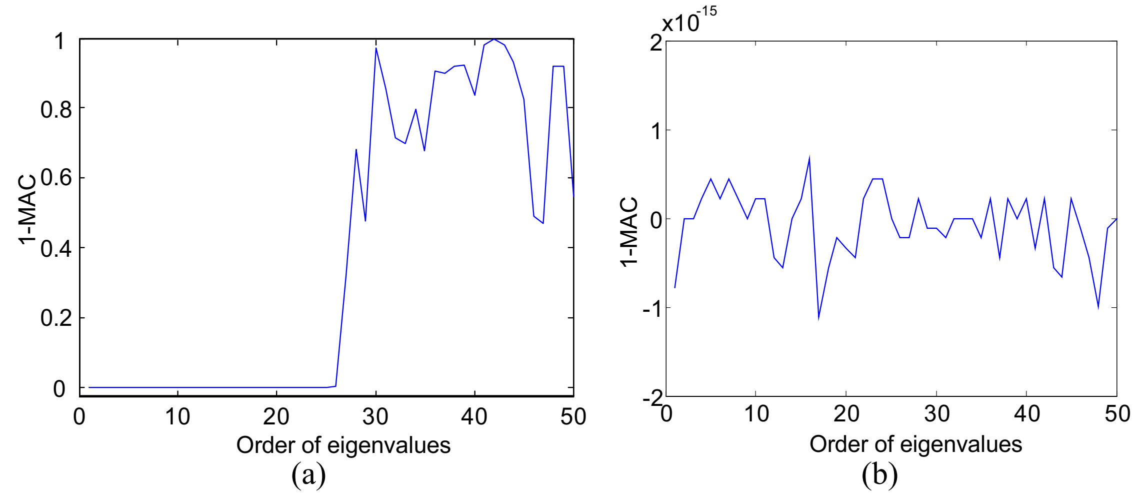


Figure 7.5. Complex eigensolver accuracy check on 50 x 50 matrix (eigenvector): (a) smaller eigenvalues; and (b) larger eigenvalues.

The matrix size for which an eigenanalysis can be performed is limited by available memory. The size of a complex double precision matrix is twice as large as that of real double precision matrix. Double precision eigenvalues and eigenvectors of an $n \times n$ complex matrix occupies $16n^2 + 16n$ B of memory. The eigensolver uses another $n \times n$ matrix internally, resulting in a need for $32n^2 + 16n$ B memory in total; the size of memory needed to keep other small internal variables are not counted. When n = 89, the RAM space occupied by these matrices exceeds the size of the 256 kB of on-board RAM on the Imote2.

7.1.4 Complex matrix inverse

A complex matrix inverse function is developed based on the Gauss-Jordan matrix inverse method described in *Numerical Recipes in C* (Press et al., 1992). The accuracy of the algorithm is examined by calculating the matrix product of the inverse and original matrices. The identity matrix was obtained as the output.

The complex matrix inverse algorithm for an $n \times n$ complex matrix needs $16n^2 + 16n$ B of memory. $16n^2$ B is first used to store the original matrix and then replaced with the matrix inverse. $16n$ B is internally used. The necessary memory space, $16n^2 + 16n$, exceeds 256 kB when $n = 126$.

7.1.5 Sort

A quick sort algorithm is developed based on *Numerical Recipes in C* (Press et al., 1992). One application example requiring two vectors be sorted simultaneously is the ordering of natural frequencies in ascending order and the corresponding reordering of multiple modal parameters. This function is implemented using floating point operations.

The memory size required to run this function is approximately $8n$ B, where n is the length of the vector to be sorted; the size of floating type variable is 4 B. Sorting of more than 30,000 data points can be achieved with 256 kB memory. This function is not considered to be expensive in terms of memory usage.

7.2 DCS implementation

Components of DCS (i.e. Sensing, NExT, ERA, DLV methods, and DCS logic) are implemented on the Imote2s. The outputs of each step of the DCS algorithm on Imote2s is compared with reference values calculated on the PC using MATLAB. Sensing is examined by comparing signals from the Imote2s with those from conventional reference accelerometers. To check the validity of the various DCS components, acceleration data is injected from the PC, instead of acceleration measured on the Imote2, and used for the numerical evaluation. The Imote2 is shown to perform DCS calculations as designed.

7.2.1 Sensing

The middleware service developed in section 5.3 is implemented on the Imote2, and the sensing capability investigated. The sensing capability is first calibrated against reference sensors. In anticipation of the damage detection experiment with Imote2s in the next chapter, the Imote2s and reference sensors are placed on the three-dimensional truss structure, and the measured acceleration signals are examined. Synchronization of the measured signals is considered by sensing acceleration responses of the truss and comparing the phase characteristics of the signals.

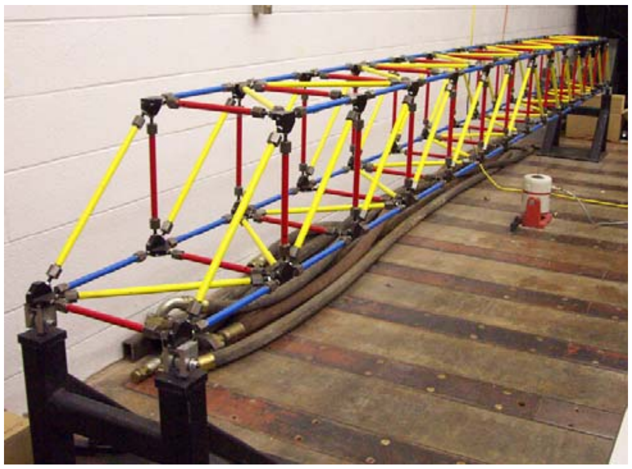


Figure 7.6. Three-dimensional, 5.6 m-long truss structure.

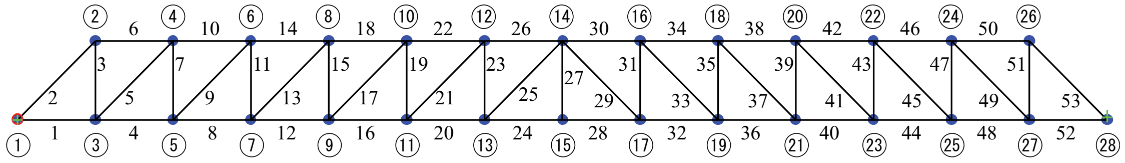


Figure 7.7. Node and element IDs of the truss.

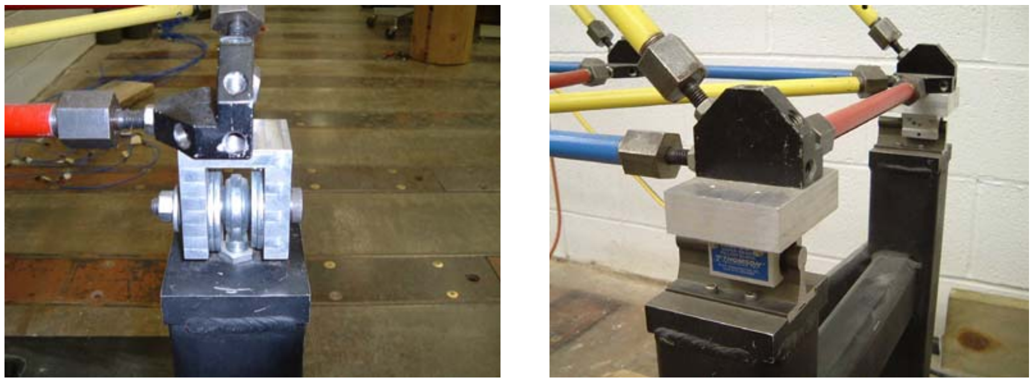


Figure 7.8. Pin and roller ends (Gao, 2005).

The structure on which the calibration is conducted is the 5.6m long, three-dimensional truss (see Figures 7.6 and 7.7) located at the Smart Structures Technology Laboratory (SSTL) of the University of Illinois at Urbana-Champaign (http://cee.uiuc.edu/sstl/). Originally designed at the Structural Dynamics and Control/Earthquake Engineering Lab in the University of Notre Dame (Clayton & Spencer, 2001), the truss has 14 bays, each of which is 0.4 m in length. The truss sits on two rigid supports. One end of the truss is pinned to the support, and the other is roller-supported (see Figure 7.8). The pinned end can rotate freely with all three translations restricted. The roller end can move in the longitudinal direction.

The truss members are steel tubes with an inner diameter of 10.9 mm and an outer diameter of 17.1 mm. The joints of the elements are specially designed so that the truss

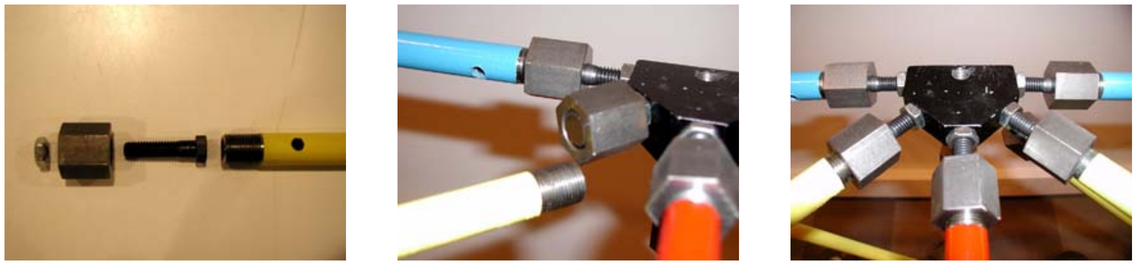


Figure 7.9. Details of the joint (Gao, 2005).

Figure 7.10. Magnetic shaker (Gao, 2005).

member can be easily removed or replaced to simulate damage without dissembling the entire structure. A detailed picture of the joint is shown in Figure 7.9. As can be seen, a truss member can be installed/removed by tightening/loosening the collars at the two ends of a member.

The truss is excited vertically by a Ling Dynamic Systems permanent magnetic V408 shaker (see Figure 7.10) that can generate a maximum force of 20 lbs. with a dynamic performance ranging from 5 to 9,000 Hz. A band-limited white noise is sent from the computer to the shaker to excite the truss structure up to 100 Hz. The shaker is connected to the bottom of the outer panel using a small steel rod. A PCB piezotronics load cell (model 208B02) is installed between the steel rod and the bottom of the joint to measure the input to the structure. This load cell has a sensitivity of 50 mV/lb, a frequency range of 0.001 to 36,000 Hz, and a measurement range of 100 lbs. in both compression and tension. In this report, input force measurement is not used for damage detection. However, the load cell is installed to confirm that the input force is approximately a band-limited white noise.

Two Imote2s are firmly attached on a node of the truss by hot glue to measure structural responses at this point in three directions (see Figure 7.11). Two Imote2s are installed next to each other; corresponding signals from the two Imote2s are expected to be identical if the truss node behaves as a rigid body without rotational motion. Even though rotational behavior of the truss node is not zero, translational motion is expected to be of much larger magnitude. The two signals should, therefore, show good agreement. Differences in the two signals are attributed primarily to observation noise specific to each

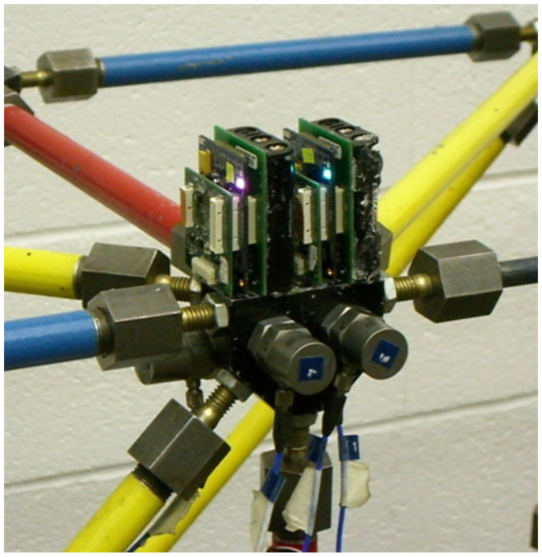


Figure 7.11. The Imote2s on a truss node.

Figure 7.12. Accelerometer: model 353B33 and magnetic base.

of the Imote2 nodes, while noise common to both Imote2 nodes needs to be detected in comparison with reference sensors.

PCB high-sensitivity piezotronics accelerometers (see Figure 7.12.) are used as reference accelerometers. These accelerometers have a sensitivity of approximately 100 mV/g, a frequency range of 1 to 4,000 Hz, and a measurement range of ±50g. These accelerometers are mounted on the structure through a magnetic base. These magnetic bases facilitate easy relocation of the accelerometers between tests.

Four PCB accelerometers are attached on the same structural node as the one that the Imote2s are attached to. Two of them measure acceleration in the longitudinal direction, while the others measure acceleration in the transverse direction. To investigate vertical acceleration measurements, the two accelerometers used in longitudinal acceleration measurement are relocated so that these sensors measure vertical acceleration. Thus, at least two reference accelerometers measure acceleration in one direction; differences in signals from these two sensors indicate noise components that are not common to both of them.

A four-channel 20-42 Siglab spectrum analyzer (Spectral Dynamics, Inc., 2007) provides the signal to the shaker and measures the inputs or reference sensor signals.

Figure 7.13. Amplifier (Gao, 2005).

Eight-pole elliptical AA filters are employed for both the input and output measurements with a cutoff frequency of 100 Hz. The sampling rate is 256 Hz. Other equipment used during the testing includes:

• Amplifier: A Sony STR-D315 amplifier (see Figure 7.13) uses the input signal from the Siglab spectrum analyzer to drive the magnetic shaker.

• Signal conditioner: PCB four-channel signal conditioners (model 441B104) have been used in this experiment.

Using the synchronized sensing middleware service, the Imote2 is programmed to capture acceleration responses of the truss. The LIS3L02DQ accelerometer (STMicroelectronics, 2007) on the Imote2 sensor board applies an AA filter and yields digital outputs. The cutoff frequency of the AA filter is fixed with respect to the sampling frequency (see Table 7.2). Because the structural modes of the truss whose natural frequencies of interest are as high as 100 Hz need to be analyzed, the sampling frequency is set as 560 Hz. After measurement, signals are resampled at 280 Hz, with the resulting usable bandwidth of the signal being a little over 100 Hz.

Table 7.2. *Cutoff and Sampling Frequencies of LIS3L02DQ Accelerometer*

Decimation factor	*Cutoff frequency (Hz)*	*Sampling frequency (Hz)*
128	*70*	*280*
64	*140*	*560*
32	*280*	*1120*
8	*1120*	*4480*

The digital lowpass filter employed in the resampling approach is designed using Matlab (The Mathworks, Inc., 2007). The passband cutoff frequency needs to be higher than the frequency range of interest, i.e. 100 Hz, while the stopband cutoff frequency is usually lower than the Nyquist frequency of the resampled signals. The filter design with these two cutoff frequencies being much lower than the sampling frequency of the upsampled signal and close to each other requires a large number of filter coefficients. To alleviate this filter design difficulty, the stopband cutoff frequency is increased. Instead of

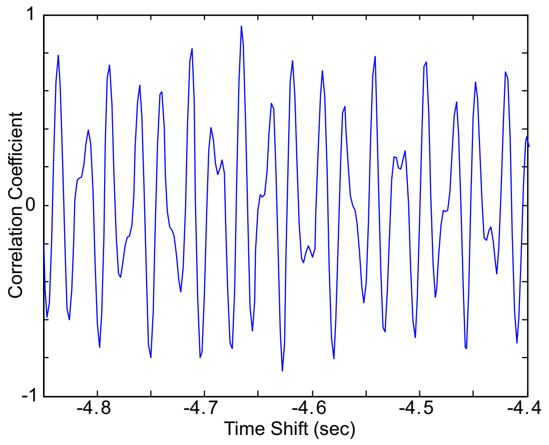
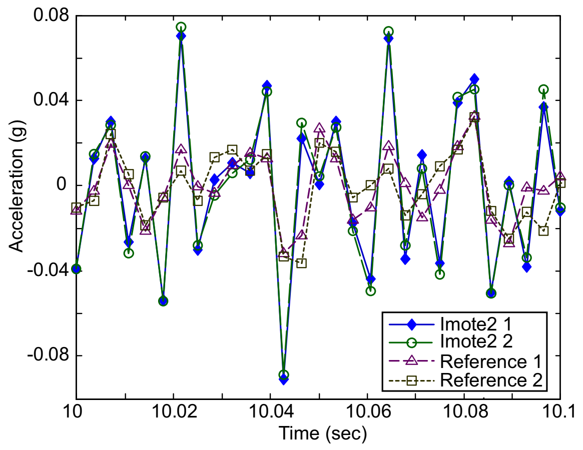


Figure 7.14. Correlation coefficients plotted against time shift.

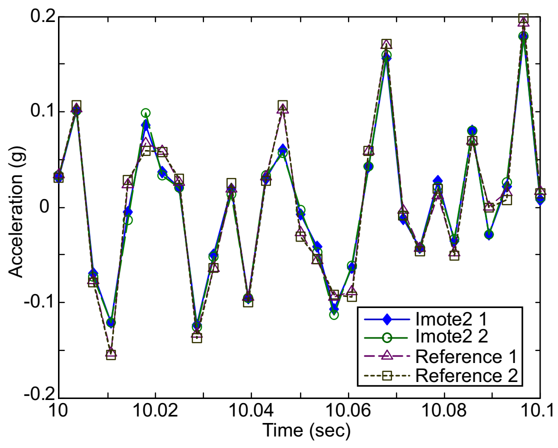
keeping the stopband cutoff frequency lower than the Nyquist frequency and completely eliminating the aliasing components, the cutoff frequency is set so that the aliasing components are above 100 Hz. For example, when sampling frequency is converted from 560 to 280 Hz with the passband cutoff frequency of 100 Hz, the stopband cutoff frequency is set to 180 Hz instead of 140 Hz. Note that the actual sampling frequency of the sensor board may have 10 percent variation and is not exactly 560 Hz. The stopband cutoff frequency normalized to the upsampled sampling rate is set to be smaller than 180/560*L,* allowing the variation in the denominator. *L* is the upsampling factor. Likewise, the normalized passband cutoff frequency is set to be larger than 100/560*L*. In this way, a filter allowing aliased components outside of the frequency range of interest is designed with fewer filter coefficients than a filter completely eliminating aliased components.

The length of the measured data is subject to the RAM size limitation. The sensing function on the Imote2 first stores data in the on-board RAM; the size of RAM limits the sensing duration. For example, a three-axis acceleration measurement using a 16-bit data type for each axis and having 11,264 data points occupies about 67 kB of RAM. Because sensing is first performed at a higher sampling frequency and then downsampled, the RAM originally occupied by the measured acceleration data is about 134 kB. 256 kB of RAM on the Imote2 does not allow continuous measurement of acceleration for an indefinite amount of time. Once the memory space becomes full, sensing is stopped, the stored data is copied to Flash memory, and sensing can be started again.

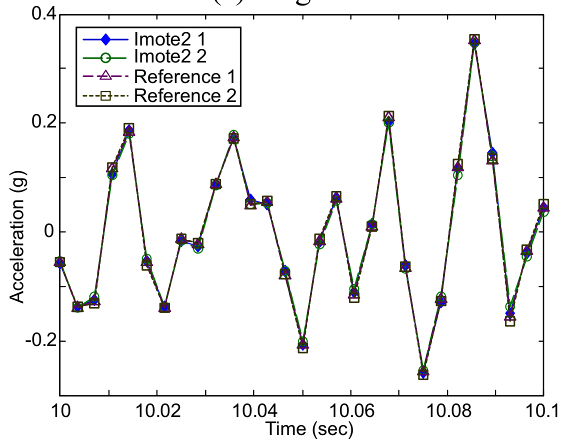
Acceleration signals from the Siglab spectrum analyzer and Imote2s are compared with each other to examine the sensing capability of the Imote2. Because the sampling frequency of the reference signals is different from that of Imote2 signals, which is set at 280 Hz, the reference signals are resampled to 280 Hz. The resampling algorithm described in section 5.3 is utilized. Also, the Siglab system is not synchronized with Imote2s, making direct comparison of the two time domain signals difficult. Signals from the two data acquisition systems are synchronized using the resampling algorithm discussed previously. To estimate the offset, the reference sensor signals are shifted on a time axis by a specified offset and the correlation coefficient between the two signals is calculated. Figure 7.14 provides a plot of the correlation coefficient against the time shift. The offset between the two sets of clocks is determined as the time shift giving the



(a) longitudinal



(b) transverse

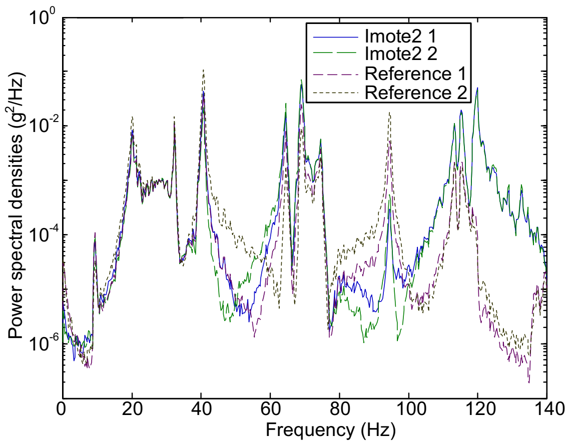


(c) vertical

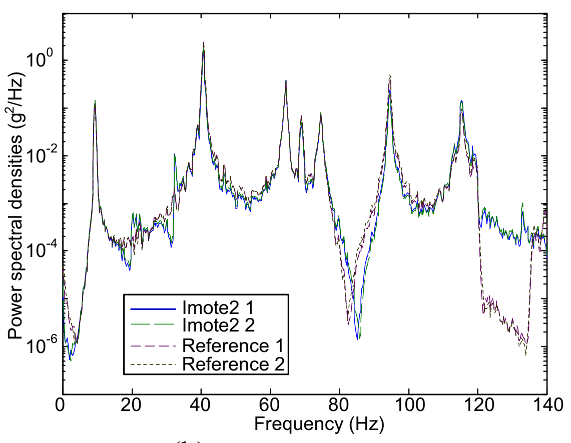
Figure 7.15. Acceleration record from Imote2s and reference sensors.

maximum absolute correlation coefficient. Note that the time shift is not restricted to multiples of the sampling period. The resampling algorithm with linear interpolation allows noninteger multiple of the sampling period as a time shift. Synchronized signals are then compared in time and frequency domains.

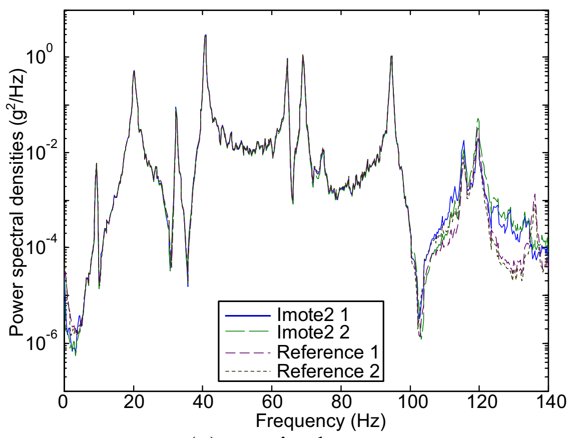
The measured signals from the Imote2s and PCB accelerometers show reasonable agreement in the time domain. Figure 7.15 shows a comparison of the four time histories. Two of them are from Imote2s and the other two are from the PCB accelerometers. The vertical acceleration signals from Imote2s and PCB accelerometers are almost identical, indicating high sensing performance of Imote2s. Transverse acceleration, however, is not as good as the vertical acceleration. Signals from PCB accelerometers and Imote2s have an observable difference, while signals from the same systems are close to each other. Longitudinal acceleration shows noticeable differences among the four signals. Here, the two signals from the PCB accelerometers show differences while the Imote2 signals appear to be the same. The noise on Imote2 signals specific to each node is considered small.



(a) longitudinal



(b) transverse

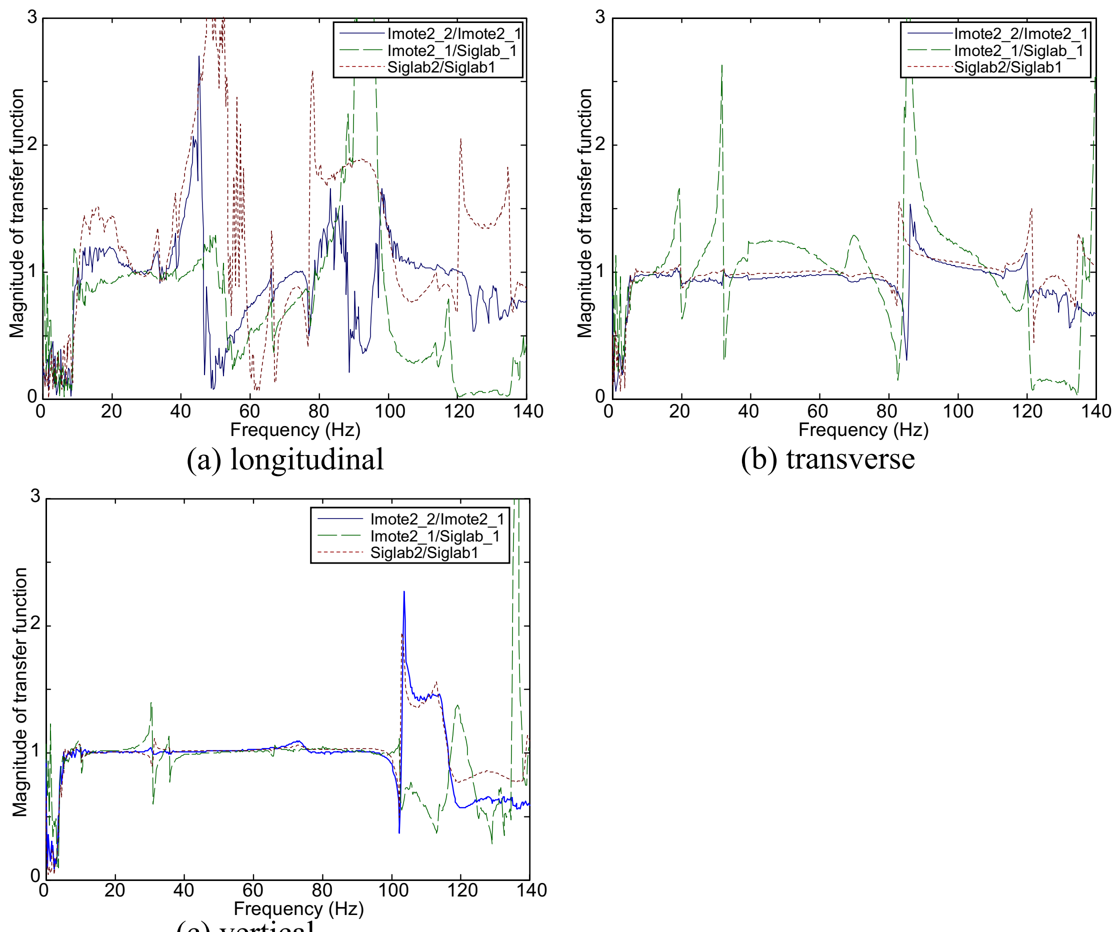


(c) vertical

Figure 7.16. Acceleration power spectral densities of Imote2s and reference sensors.

The same observation is made when the signals are compared in terms of power spectral density (see Figure. 7.16). Note that signals above 100 Hz cannot be directly comparable because the cutoff frequencies of filters involved are set around 100 Hz. As with the time domain comparison, the power spectral density of the longitudinal accelerations shows differences between the two Imote2 signals. The observable difference in the signals is not constant over frequency. Another finding from the plot for the longitudinal acceleration is that Imote2s gives smaller signal magnitude above 80 Hz than PCB accelerometers. In general, the Imote2 signals are close to each other.

Transfer functions between the Imote2 accelerometers, between reference accelerometers, and between the Imote2 and reference accelerometers are then estimated (see Figure 7.17). The transfer function magnitude plotted against frequency reveals differences in the sensitivities of two signals as a function of frequency. As can be seen in Figure 7.17, the transfer function magnitude for the vertical acceleration is nearly flat below 100 Hz. The compared sets of signals are considered close to each other confirming findings from Figures 7.15 and 7.16. The fluctuation found around 30 Hz, as well as fluctuation in low-frequency range, is considered to be due to the signals being extremely small, as shown in Figure 7.16. Transfer functions for the transverse acceleration confirm



(a) longitudinal

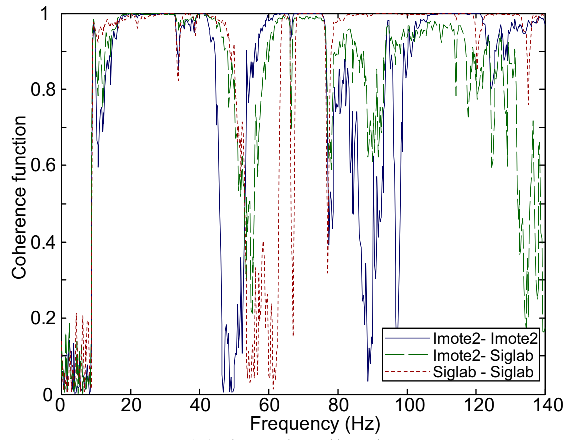
(b) transverse

(c) vertical

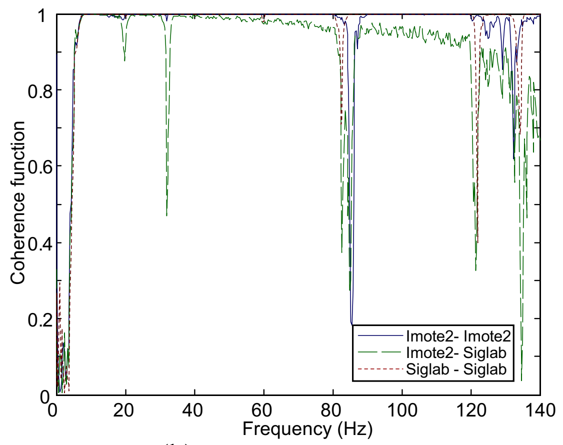
Figure 7.17. Magnitude of transfer function among Imote2s and reference sensors.

that signals from the same data acquisition system are close to each other, but signals from the Imote2 and the Siglab spectrum analyzer have noticeable differences. The transfer functions for the longitudinal acceleration fluctuate considerably, indicating that the four signals have discrepancies. In terms of the transfer functions, the longitudinal acceleration measurements seem to be noisy, though the problem is not only specific to Imote2s but also for the PCB accelerometers.

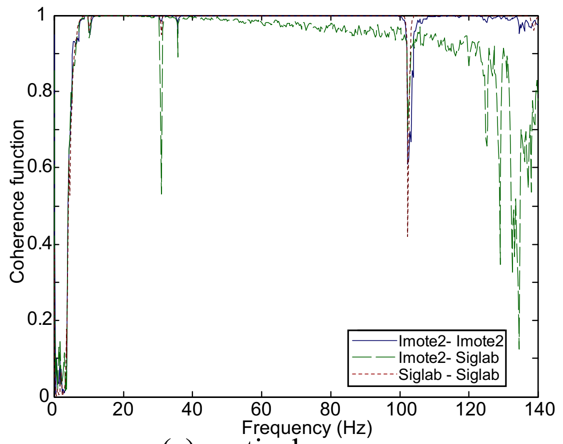
Imote2 and PCB accelerometer signals are also compared in terms of coherence (see Figure 7.18). While the transfer function magnitude can detect frequency-dependent amplification factors in two signals, the coherence function indicates the degree of linearity between two signals. Signals that are linearly related result in a coherence function that is one over all frequencies. Small values in the coherence function, on the other hand, suggest nonlinearity or noisy measurements in the corresponding frequency range. Vertical and transverse accelerations show coherence functions close to one. The gradual drop of the coherence function between an Imote2 and PCB accelerometer signals may indicate imperfect time synchronization between the two signals. Coherence functions for the longitudinal acceleration are small in some frequency ranges. The



(a) longitudinal



(b) transverse



(c) vertical

Figure 7.18. Coherence functions among Imote2s and reference sensors.

coherence function between the Imote2s is smaller than that between PCB accelerometers around 50 Hz and above 80 Hz.

To investigate the sensitivity of the other axes of the Imote2 accelerometer, the orientation of the Imote2 is changed. The vertical acceleration again seems to be accurate, while the longitudinal acceleration shows noticeable differences among the signals. The transverse acceleration measurements are not as accurate as the vertical acceleration measurements. Similar to the previous experiment, the vertical acceleration is accurate and the longitudinal measurement seems noisy. Therefore, the accuracy of the measured accelerations does not appear to depend on the specific channel used on the Imote2.

There are several possible explanations for why the vertical acceleration is accurate, while longitudinal acceleration appears noisy. First, consider the magnitude of the accelerations. Because the truss's motion is relatively small in the longitudinal direction (see Figures 7.15 and 7.16), the signal-to-noise ratio (SNR) is small for longitudinal measurements. However, similar experiments with smaller excitation also show that the vertical acceleration is accurate even when signal's strength is small; small SNR alone cannot explain why the longitudinal measurements are noisy. Another possible

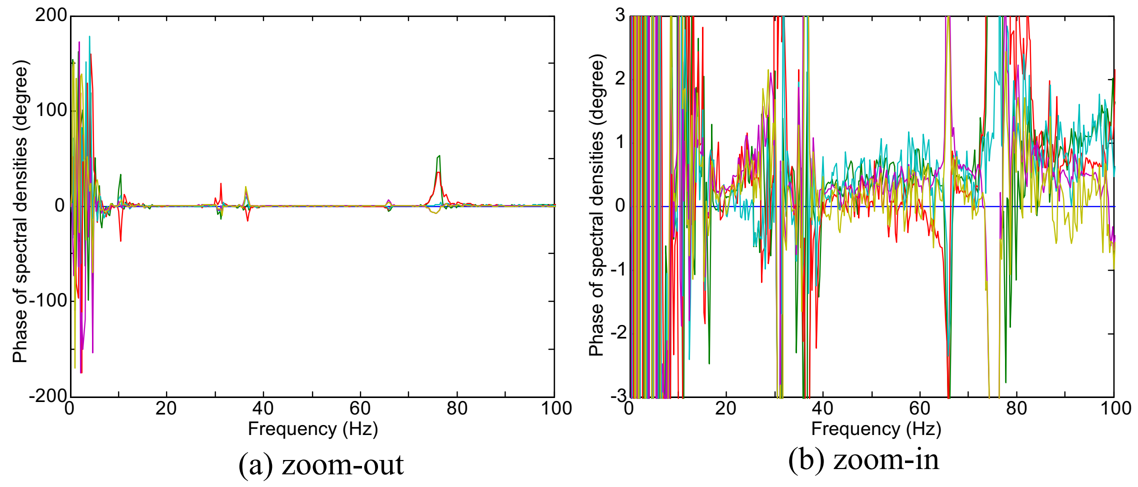


Figure 7.19. Phase of cross spectral densities.

explanation concerns crosstalk. The crosstalk specification for the Imote2 accelerometer is a maximum of 2 percent, while that of the PCB accelerometer is 5 percent. Because vertical acceleration is much larger than longitudinal acceleration, measured acceleration signals for the longitudinal direction may contain substantial crosstalk. If the actual value of the crosstalk varies from sensor to sensor within the maximum bounds, the measured longitudinal acceleration may differ from sensor to sensor; thus, each sensor may have a different contribution from the acceleration in the perpendicular axes. Further investigation may confirm this conjecture.

Time synchronization of measured signals is now examined. Six Imote2s are placed on structural nodes to measure acceleration response. Phase of cross-spectral densities among these signals is considered to indicate the accuracy of time synchronization. Synchronized signals are expected to have a phase of zero, while synchronization error results in linear phase. The larger the error is, the steeper the slope of phase is. Figure 7.19 shows the phase of the cross-spectral densities between the reference node signal and signals from the other five nodes. The phase is almost flat over the frequency range below 100 Hz. Theoretically, the phase is one degree at 100 Hz if the signals have a synchronization error of 28 μs. The phase indicates that the synchronization of the measured signals is approximately 30 μs.

7.2.2 NExT

Natural Excitation Technique (NExT) is implemented on the Imote2 using the model-based data aggregation strategy described in section 5.1.2. The validity of this implementation is examined by analyzing the same set of data both on a group of Imote2s and on the PC.

The data to be analyzed is the acceleration response of the three-dimensional truss structure. The data analyzed by Gao (2005) is converted to integers by shifting, scaling, and rounding. Assuming the usage of a 16-bit ADC with approximately 12 effective bits, the conversion maps the maximum data value to about 1,500 and the minimum value to

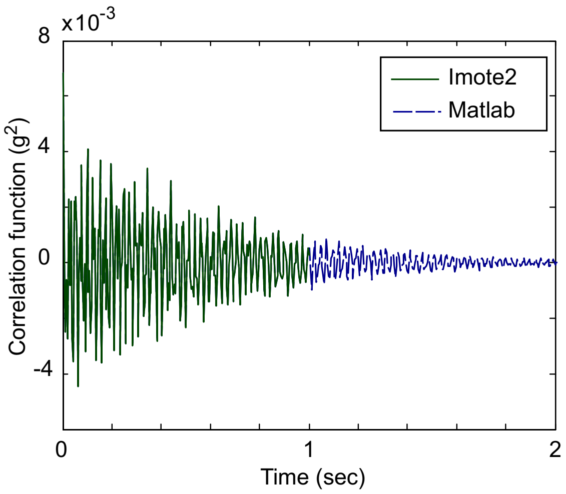


Figure 7.20. Correlation function estimates.

about -1,500. Note that the full range of a 12-bit signed integer is from -2,048 to 2,047. The integer data is then injected to the group of Imote2s using the reliable communication protocol for 16-bit integer data type described in section 5.2.3.

The acceleration response data is first split into several blocks and then injected. A block of 11,264 data points is injected to a node at a time, and this injection is repeated three times to each node. Eventually each node receives 33,792 data points per sensing channel. Dividing time histories into several blocks is incorporated to simulate the Imote2 sensing process, which must be performed repeatedly, as explained in section 7.2.1, to collect long records of data.

NExT is performed on the data injected to the Imote2s corresponding to nodes 2 to 7 (see Figure 7.7). Node 4 is designated as the cluster head, organizing distributed correlation function estimation. The number of data points used in the FFT calculations is 1,024, resulting in 21 time averages in estimation of the spectral densities for the 11,264 point data records and 63 time averages for the 33,792 point data records. Then the spectral densities are converted to correlation functions by the inverse FFT. Figure 7.20 shows the cross-correlation function between the vertical acceleration responses at nodes 3 and 4 (see Figure 7.7) calculated on the Imote2 as well as that calculated on the PC. The estimate from the PC is overlaid by the estimate from the Imote2. Note that only the first 256 data points, or one second, of the correlation function estimate on the Imote2 is reported to the base station and the cluster head to reduce communication requirements. The subsequent modal analysis using ERA usually utilizes only a portion of the correlation function, eliminating the need to transfer the entire correlation function. Figure 7.21 shows the difference between a correlation function estimated on the Imote2 and the corresponding correlation function estimated on the PC. From these figures, NExT on the Imote2 is considered to be numerically equal to that on the PC with the precision of a double data type.

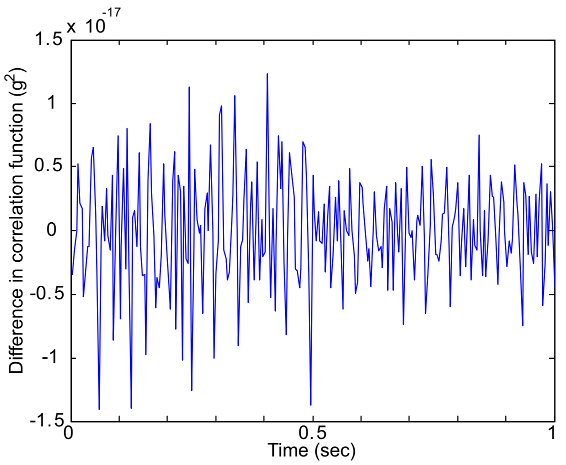


Figure 7.21. Difference in correlation function estimates.

7.2.3 ERA

The ERA method is implemented on the Imote2 using the previously developed eigensolver and SVD functions. To study the validity of the implementation, the outputs from ERA on the Imote2 are compared with those calculated on the PC.

While the algorithm is well-established as described in section 6.2, the algorithm implementation on Imote2 is subject to memory limitations, necessitating cautious consideration. One of the numerical operations requiring a large amount of memory is the SVD of the Hankel matrix as shown in Eq. (6.9) . The number of columns and rows is often in the hundreds. A Hankel matrix with more than a thousand rows is not uncommon. The Imote2's 256 kB of on-board RAM, however, is filled to capacity by a double precision matrix of size 180×180. SVD of a matrix outputs two unitary matrices and a vector containing the singular values, further limiting the size of the matrix to be analyzed with ERA. In practice, some of the memory is already used by other variables, other programs, and the OS. Therefore, the size of the Hankel matrix on the Imote2 is limited by the available memory.

A 48×50 Hankel matrix, occupying about 19 kB of RAM, is employed for implementation of ERA on the Imote2. In general, a small Hankel matrix results in the inclusion of more noise modes in the modal identification. To avoid missing physical modes in modal identification, the number of nonzero singular values in Eq. (6.9) is assumed to be larger than the expected number of physical modes. Envisioning identification of the truss vibration modes in the frequency range lower than 100 Hz, the ERA implementation assumes 28 singular values as nonzero. Among 14 identified pairs of modes, those with a small modal amplitude coherence (Juang & Pappa, 1985) or with a large damping ratio are considered to be noise and are eliminated. Even after this noise mode elimination, modes that do not show clear peaks on the cross-spectral density plots may still be present in the identified sets of modes. The Imote2 first estimates the peaks of the cross-spectral density amplitudes as local maximums, and picks the ERA output modes that have natural frequencies close to these peaks as the physical modes. ERA is, thus, implemented on the Imote2 having limited memory.

The accuracy of the developed ERA routine is now numerically examined. ERA is applied to the correlation functions estimated in section 7.2.2. The Imote2 corresponding to node 4 performs ERA and identifies the natural frequencies, damping ratios, mode shapes, modal amplitude coherences, and initial modal amplitudes. Natural frequencies, damping ratios, and mode shapes are compared with those calculated on the PC and are summarized in Table 7.3. The error in the frequency and damping ratio estimates is calculated as the difference between the estimates on the Imote2 and the PC. The estimation error in the mode shapes are investigated in terms of the Modal Assurance Criterion (MAC) defined in Eq. (7.1). As is seen in Table 7.3, the modal identification results on the Imote2 and those on the PC are identical with the precision of double data type.

Table 7.3. *Identification of Natural Frequencies, Damping Ratios and Mode Shapes*

	Natural Frequency		*Damping Ratio*		*Mode Shape*
mode	*f (Hz)*	*difference* (10^{-12})	ζ *(%)*	*difference* (10^{-12})	*1-MAC* (10^{-16})
1	20.1526	5.3078	3.5523	2.3794	2.2204
2	41.2039	-0.4263	0.3439	0.1252	0
3	61.9044	-0.1137	0.3812	-0.0944	0
4	66.9313	-0.0142	0.5763	-0.0648	-2.2204
5	70.4128	-0.5826	0.7909	-0.9630	2.2204
6	93.8481	-0.0426	0.3118	0.7626	-2.2204

7.2.4 DLV methods

Implementation of the DLV methods on the Imote2 is described in this section. These implementations are validated through comparison between the results on the Imote2 and the PC.

Deliberate consideration is given to the implementation of the DLV methods to accommodate the limited hardware resources on the Imote2. One of the numerical operations requiring a large amount of memory and CPU time is stress analysis under the DLV loading. If an FEM model of the entire structure needs to be stored on a smart sensor node, the model may exceed available memory. Numerical operations involved in the analysis of the FEM model, i.e., calculation of static displacements and stresses under the DLV loading, are not trivial. Instead, the linear nature of the analysis is used to simplify the process. A matrix to convert input force to stress is calculated in advance and injected to the cluster heads; rather than to run the entire structural analysis, the cluster heads simply need to compute the product of the matrix and DLVs. Furthermore, the conversion matrix needs to keep only the submatrix corresponding to the structural node and elements in the local sensor community corresponding to the cluster head. The submatrix converts input forces applied at nodes in the local sensor community to stress in elements in the same community. The structural analysis is, thus, performed as the product of the modest size matrix and DLVs

The DLV method implementation on the Imote2 is now numerically examined. The DLV method based on a known mass perturbation is employed in this numerical validation; a similar procedure was employed to implement and validate the SDLV method.

Prior to monitoring local sensor communities, mass normalization constants are estimated. These constants are also estimated on the Imote2. The structural responses of the truss before and after the addition of the known mass at node 11 are injected to 10 Imote2s corresponding to nodes 7, 9, 11, 13, 15, 17, 19, 21, 23, and 25. The Imote2 at node 21 works as the cluster head and applies NExT and ERA before and after the mass perturbation. Using the identified modal parameters, the cluster head estimates the mass normalization constants using Eq. (6.21). The estimated mass normalization constants are listed in Table 7.4 together with the difference between the constants estimated on the Imote2 and the PC. The mass normalization constant estimation on the Imote2 is numerically identical to that on the PC.

Monitoring of the local sensor community consisting of six Imote2s at nodes 2, 3, 4, 5, 6, and 7 is now examined. The injection of data, correlation function estimates by NExT, and modal identification by ERA are repeated for the truss acceleration response data measured before and after element 9 is replaced with a thinner element. Identified modal frequencies and mode shapes before and after damage, as well as the estimated mass normalization constants, are used to construct flexibility matrices. As intermediate results, the singular values in Eq. (6.24) estimated on the Imote2 and on the PC are compared in Table 7.5. The DLVs calculated on the Imote2 are also compared with those estimated on the PC. The singular values and DLVs are considered numerically identical. The DLVs are then applied to the truss model to estimate the normalized accumulated stress. As shown in Table 7.6, the stress in element 9 is smaller than the threshold value of 0.3. Because the normalized accumulated stress is only compared with the threshold value, the stress does not need to have many effective bits. Therefore, the stress estimation on the Imote2 is performed in a single-precision float data type having approximately seven effective digits. The normalized accumulated stress estimated on the Imote2 and on the PC is identical with respect to the precision of the float data type.

7.2.5 DCS logic

The DCS logic to find inconsistency in damage localization results is implemented on the Imote2. Sensor communities close to each other exchange information about damaged elements. Because the topology of sensor communities is formulated to have overlaps in monitored elements, at least two sensor communities monitor each element of the structure. If some communities find contradicting damage detection results, these communities repeat sensing, data processing, and the DCS logic. Also, if sensor communities detect damaged elements in a consistent manner, these communities report

Table 7.4. *Mass Normalization Constant Estimation*

Mode	*Frequency*	*Mass Normalization Constant*	*Difference in Mass Normalization Constant* (10^{-14})
1	*19.9078*	*0.011098*	*0.3530*
2	*41.1663*	*0.007703*	*1.0122*
3	*62.6000*	*0.008077*	*0.0298*
4	*67.3655*	*0.006587*	*-0.2212*
5	*71.4202*	*0.003260*	*-1.7906*
6	*94.2266*	*0.011514*	*0.7700*

Table 7.5. *Summary of SVD in the DLV Method*

Singular Values		*DLVs*
Values	*Difference* (10^{-18})	*1-MAC* (10^{-16})
2.8380×10^{-14}	0.4224	0
2.6216×10^{-13}	0.5740	-2.2204
1.9715×10^{-12}	4.4483	-2.2204
4.2358×10^{-12}	0.0432	-4.4409
1.3588×10^{-11}	2.9774	-2.2204
3.0088×10^{-11}	-1.7915	-2.2204
5.4073×10^{-11}	1.8335	0
6.8856×10^{-11}	1.2435	0
1.2346×10^{-9}	4.0434	2.2204
1.6543×10^{-9}	-3.6429	2.2204
2.9298×10^{-8}	4.1185	0
1.1808×10^{-7}	3.4197	0

the damaged elements to the base station. The details of the DCS logic are described herein with reference to Figure 7.22.

When the manager sensor, which also works as the cluster head of a sensor community, completes application of the DLV method, the manager initializes itself to apply the DCS logic and sends a command to the next cluster head. For explanation purposes, the cluster heads are assumed to be arranged as shown in Figure 7.22. Cluster

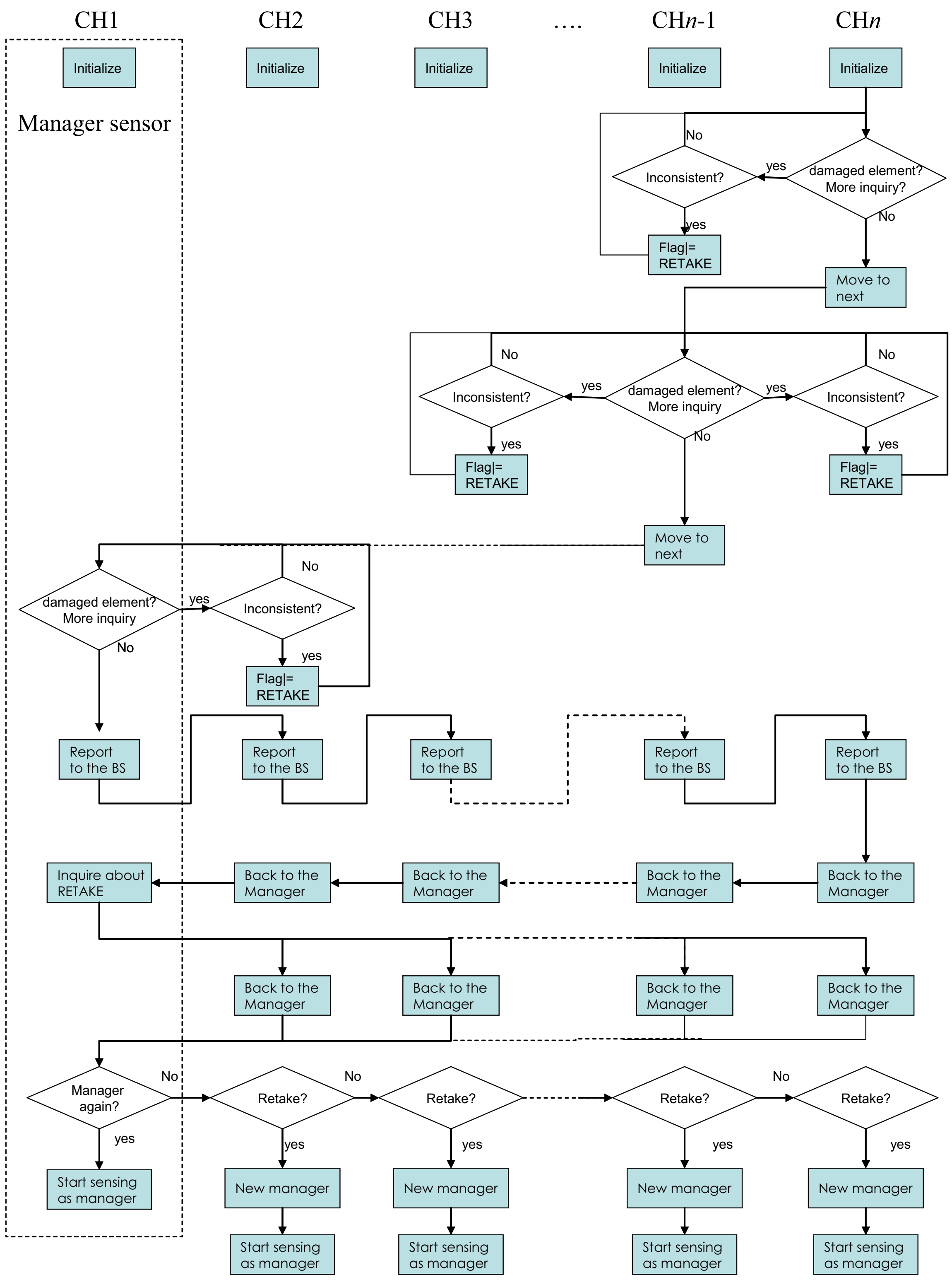


Figure 7.22. DCS logic.

Table 7.6. *Normalized Accumulated Stress Estimation*

Element ID	*Normalized Accumulated Stress*	*Difference* (10^{-8})
3	0.6377	-2.1416
4	0.6662	0.3378
5	0.9751	4.9952
6	0.4506	-2.5601
7	0.3470	1.9430
8	0.6078	-5.3461
9	0.1368	0.2829
10	1.0000	0
11	0.6226	0.2349

configuration. However, cluster heads do not need to physically line up; rather they need to line up logically. In Figure 7.22, the manager sensor, CH1, notifies the cluster head to the right, CH2. The receiver cluster head also initializes itself and notifies the next cluster head. The initialization and notification are repeated until no cluster head to the right is found.

The right-most cluster head then starts the procedure of finding inconsistent damage detection results. This cluster head first checks whether damage is detected in its sensor community. If not detected, the operation moves to the left cluster head. If detected, the cluster head inquires with the neighboring cluster heads whether the damage detection results are consistent. If inconsistent, a flag on the cluster heads involved in the transaction are marked as “RETAKE”. The search for inconsistency then moves to the left cluster head, where the procedure is repeated. This inconsistency search and transition to the left node are repeated until the process reaches the left-most cluster head.

When the left-most cluster head finishes the inconsistency search, the process of reporting to the base station begins. The left-most cluster head reports damage detection results, i.e. damaged element ID and any inconsistencies, to the base station. After this cluster head finishes reporting, the operation moves to the right cluster head. This procedure is repeated until all of the cluster heads report their results. Then, the operation moves back to the manager sensor. While all of the nodes are designed to report DCS logic results to the base station for debugging purpose, the code can be easily changed so that only cluster heads with damaged elements report to the base station.

The manager sensor then inquires of the other cluster heads whether local sensor communities organized by these cluster heads need to retake data. Only local sensor communities with inconsistent damage detection results participate in the next round of DCS. If the manager sensor in the previous round of DCS does not participate in the next round, one of the cluster heads takes over the manager sensor’s role.

```
 1
 2  Node ID 102 RETAKE 1
 3
 4  INCONSISTENT
 5
 6  Inconsistent Element ID: 8
 7
 8  ID 102 # of Damaged Element: 0
 9
10
11  Node ID 91 RETAKE 1
12
13  INCONSISTENT
14
15  Inconsistent Element ID: 8
16
17  ID 91 # of Damaged Element: 1
18
19  Damaged Element ID: 8
```

Figure 7.23. Report on damage elements.

This DCS logic is implemented on Imote2s and tested assuming acceleration measurement of the three-dimensional truss structure. Two Imote2s corresponding to cluster heads at nodes 4 and 6 receive simulated damaged element information from the PC. These two Imote2s then apply the DCS logic. Figure 7.23 shows the report from these two cluster heads after application of the DCS logic. In this case, the cluster head at node 4 is assumed not to have detected damage, while the cluster head at node 6 is supposed to have detected damage at element 8, as indicated in lines 8, 17, and 19. These two nodes find inconsistency and decide to retake data, as indicated by the RETAKE flag on lines 2 and 11. Node IDs on lines 2 and 11 are sensor node IDs unique to each Imote2 and different from the truss's structural node IDs. The DCS logic implementation on the Imote2 is, thus, demonstrated.

7.2.6 Final implementation on the Imote2

With the developed components of DCS for SHM, the entire strategy is realized on Imote2s. Simple flow charts of the process are shown on Figures 7.24 and 7.25. DCS for SHM involves at least four sets of measurements: (a) measurement over the whole structure without mass perturbation for mass normalization constant estimation, (b) measurement over the whole structure with mass perturbation for mass normalization constant estimation, (c) measurement in a local sensor communities to obtain the base line flexibility matrix, $\mathbf{F_u}$, and (d) measurement in a local sensor communities for monitoring. All of these measurements require similar procedures, with the differences being: participating sensor members, the parameters injected at the beginning, and the numerical calculation applied to the ERA results.

Extension of DCS for SHM using the SDLV, as explained in section 6.6, simplifies these steps. The SDLV does not require estimation of mass normalization constants, eliminating the need for the first two sets of measurement over the entire structure. The third and fourth sets of measurements are the same as the original DCS for SHM, with the exception being numerical operations in the DLV method.

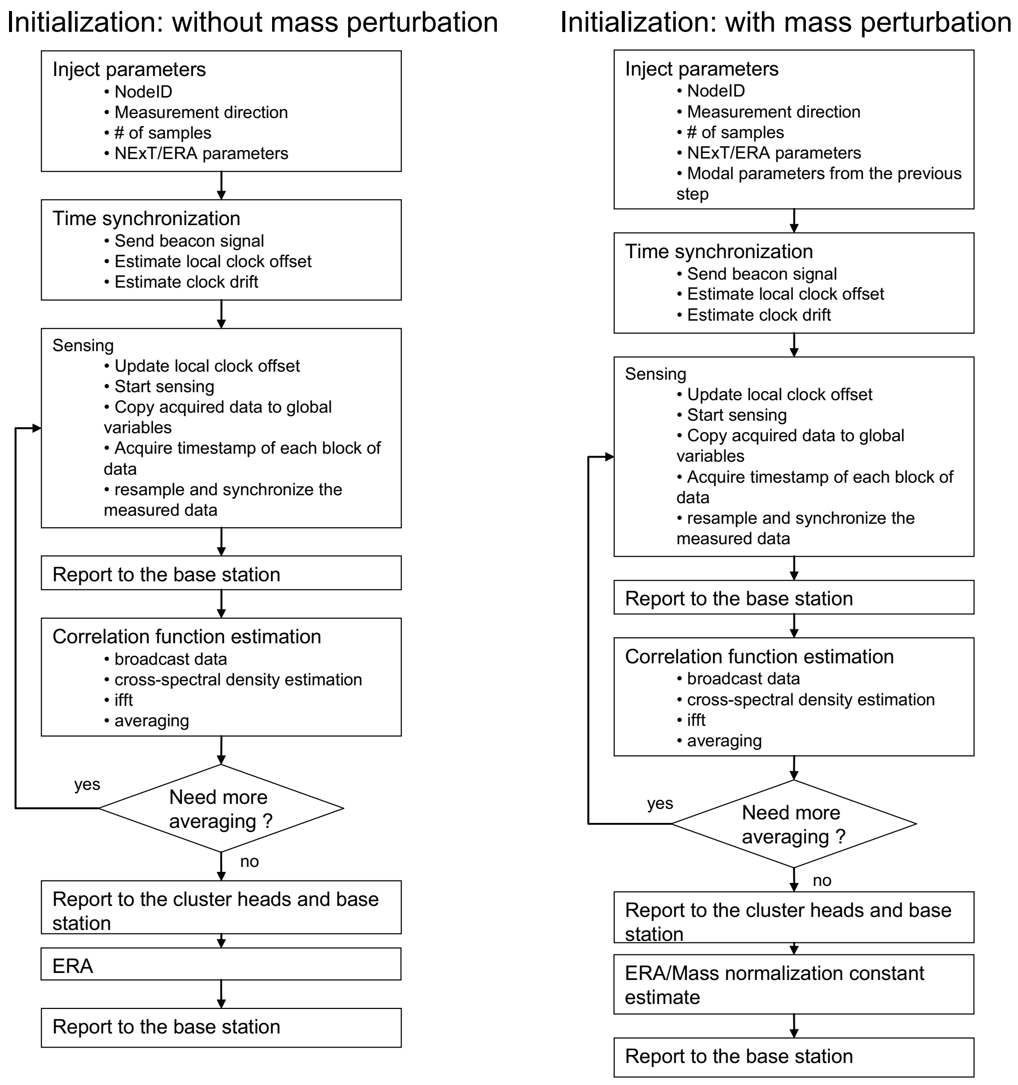

Initialization: without mass perturbation
Inject parameters
• NodeID
• Measurement direction
• # of samples
• NExT/ERA parameters
Time synchronization
• Send beacon signal
• Estimate local clock offset
• Estimate clock drift
Sensing
• Update local clock offset
• Start sensing
• Copy acquired data to global variables
• Acquire timestamp of each block of data
• resample and synchronize the measured data
Report to the base station
Correlation function estimation
• broadcast data
• cross-spectral density estimation
• ifft
• averaging
yes
Need more averaging ?
no
Report to the cluster heads and base station
ERA
Report to the base station
Initialization: with mass perturbation
Inject parameters
• NodeID
• Measurement direction
• # of samples
• NExT/ERA parameters
• Modal parameters from the previous step
Time synchronization
• Send beacon signal
• Estimate local clock offset
• Estimate clock drift
Sensing
• Update local clock offset
• Start sensing
• Copy acquired data to global variables
• Acquire timestamp of each block of data
• resample and synchronize the measured data
Report to the base station
Correlation function estimation
• broadcast data
• cross-spectral density estimation
• ifft
• averaging
yes
Need more averaging ?
no
Report to the cluster heads and base station
ERA/Mass normalization constant estimate
Report to the base station

Figure 7.24. The block diagram of mass normalization constant estimation.

At the beginning, parameters such as node IDs of the leaf nodes and measurement directions are injected to the manager sensor and cluster head nodes. Also, these nodes are notified which of the four sets of measurements shown in Figures 7.24 and 7.25 needs to be performed. Information from the base station or users is transferred only at this stage. It is considered possible to inject these parameters only once and store them on nonvolatile memory. In this way, networks of smart sensors can work as autonomous stand-alone systems.

At the end of parameter injection, time synchronization begins. To increase the possibility of successful time synchronization, and to estimate clock drift, time synchronization is repeated multiple times. Time synchronization is also performed later between measurements.

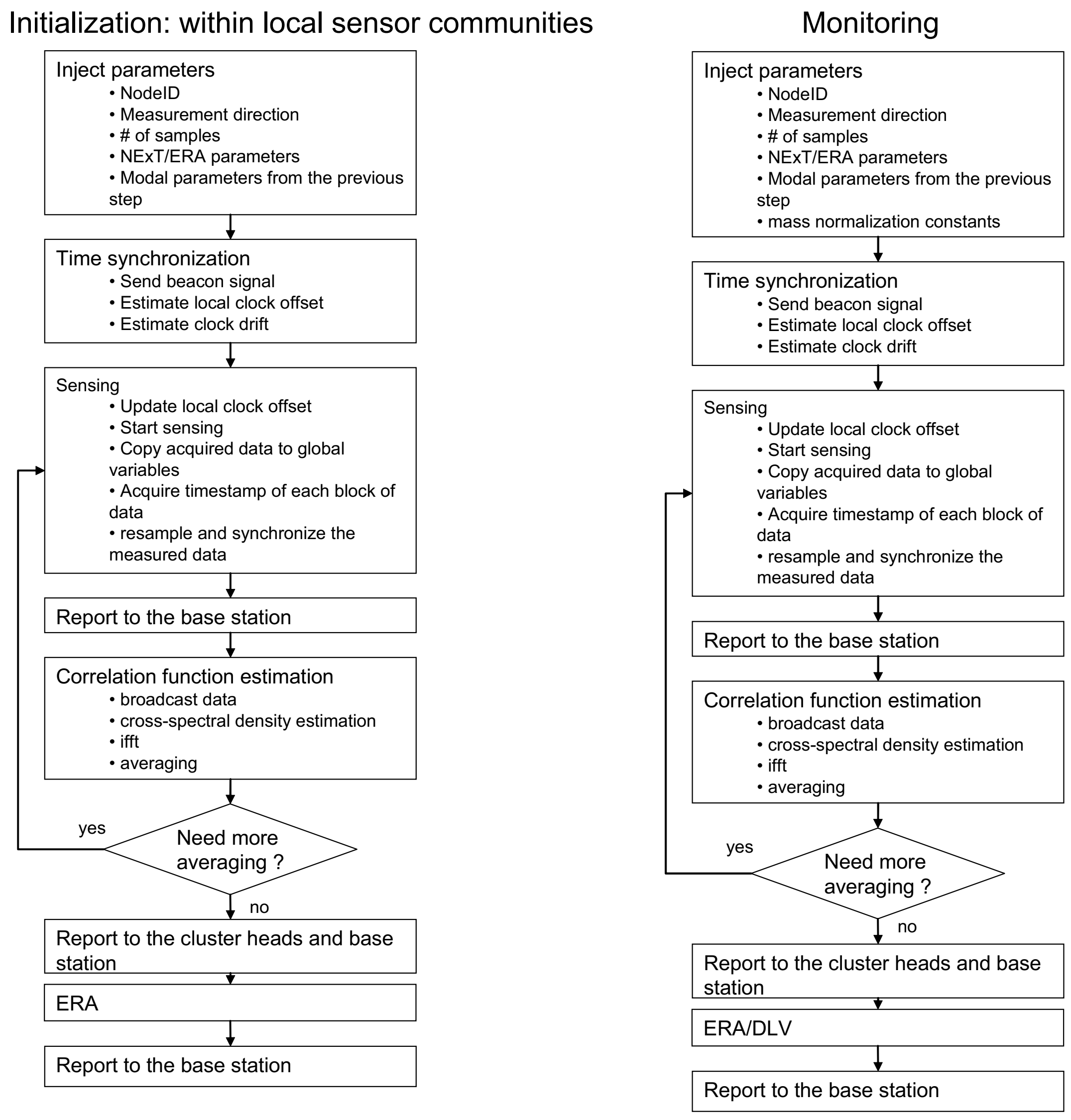

Initialization: within local sensor communities
Inject parameters
• NodeID
• Measurement direction
• # of samples
• NExT/ERA parameters
• Modal parameters from the previous step
Time synchronization
• Send beacon signal
• Estimate local clock offset
• Estimate clock drift
Sensing
• Update local clock offset
• Start sensing
• Copy acquired data to global variables
• Acquire timestamp of each block of data
• resample and synchronize the measured data
Report to the base station
Correlation function estimation
• broadcast data
• cross-spectral density estimation
• ifft
• averaging
Need more averaging ?
yes
no
Report to the cluster heads and base station
ERA
Report to the base station
Monitoring
Inject parameters
• NodeID
• Measurement direction
• # of samples
• NExT/ERA parameters
• Modal parameters from the previous step
• mass normalization constants
Time synchronization
• Send beacon signal
• Estimate local clock offset
• Estimate clock drift
Sensing
• Update local clock offset
• Start sensing
• Copy acquired data to global variables
• Acquire timestamp of each block of data
• resample and synchronize the measured data
Report to the base station
Correlation function estimation
• broadcast data
• cross-spectral density estimation
• ifft
• averaging
Need more averaging ?
yes
no
Report to the cluster heads and base station
ERA/DLV
Report to the base station

Figure 7.25. The block diagram of monitoring in a sensor community.

Synchronized sensing is performed in the manner described in section 5.3. The number of sensing samples is set to be 11,264. The raw data is sent back to the base station for debugging purposes. This reporting allows numerical operations on Imote2s to be reproduced on the PC; the validity of data processing on the Imote2 can also be examined. Centrally collecting all of the data takes a substantial amount of time and can be eliminated once this DCS system is fully developed.

Correlation functions are then estimated in a distributed manner. The model-based data aggregation approach explained in section 5.1 is employed. If the number of averages in correlation function estimation, n_d in Eq. (5.1), is smaller than the predetermined value, sensing, reporting to the base station, and correlation function estimation are repeated (see Figures 7.24 and 7.25).

The estimated correlation functions are reported back to the cluster heads and base station. While cluster heads need correlation function estimates from their leaf nodes, the base station does not need the estimates except for the debugging purposes. Reporting to the base station can be omitted once the system is fully developed.

Cluster heads then apply ERA to the correlation function estimates. Natural frequencies and mode shapes are calculated. Based on these modal parameters, mass normalization constants are estimated, or damage localization is performed. During monitoring in the local sensor communities, the DCS logic is applied to the outcome of the DLV procedure. The results are then sent back to the base station. Some of these results are injected to cluster heads at the beginning of the next set of measurement. For example, the estimated mass normalization constants are injected to the cluster heads at the beginning of the monitoring process by the local sensor communities. This reporting is also not necessary, once the system is fully developed. The outcome of one set of measurements can be kept on RAM or nonvolatile memory for the next set of measurements, eliminating the need for the parameter injection.

In the implementation of DCS for SHM, the frequency scalable feature of the Imote2 is utilized. The microprocessor on the Imote2 can operate at multiple frequencies. Because the driver on TinyOS supports only the 13 and 100 MHz modes at the time, the operational frequency is switched between only these two values. Numerical calculations in NExT, ERA, and DLV, as well as sensing, are all performed at 100 MHz. Other tasks such as communication are performed at 13 MHz.

All of the tasks in DCS for SHM need to be performed in the proper order by relevant nodes. A 1-Byte variable containing instruction for the next task is utilized in organizing all of the tasks. All of the communication packets have 1-byte variable to describe the instructions. At the end of transmission and reception, this instruction is processed in a task named “ProcessInstruction”. “ProcessInstruction” then looks for the block of codes following the “switch” statement corresponding to the instruction value. Operations described in the block of codes are executed. At the end of the block, new values can be assigned to the instruction byte to execute the next task, or a communication packet carrying new values for the instruction byte can be sent out. Components of DCS for SHM explained in this chapter are synthesized in this manner. Table 7.7 summarizes the instructions, associated operations, and information accompanying the communication packet. The implementation on the Imote2 is summarized from Figures 7.26 to 7.30.

7.3 Summary

This chapter described the implementation of DCS for SHM on a network of Imote2s. The validity of the system is numerically examined in a component-by-component manner. In the next chapter, the Imote2s are installed on a three-dimensional truss and the algorithms are experimentally verified.

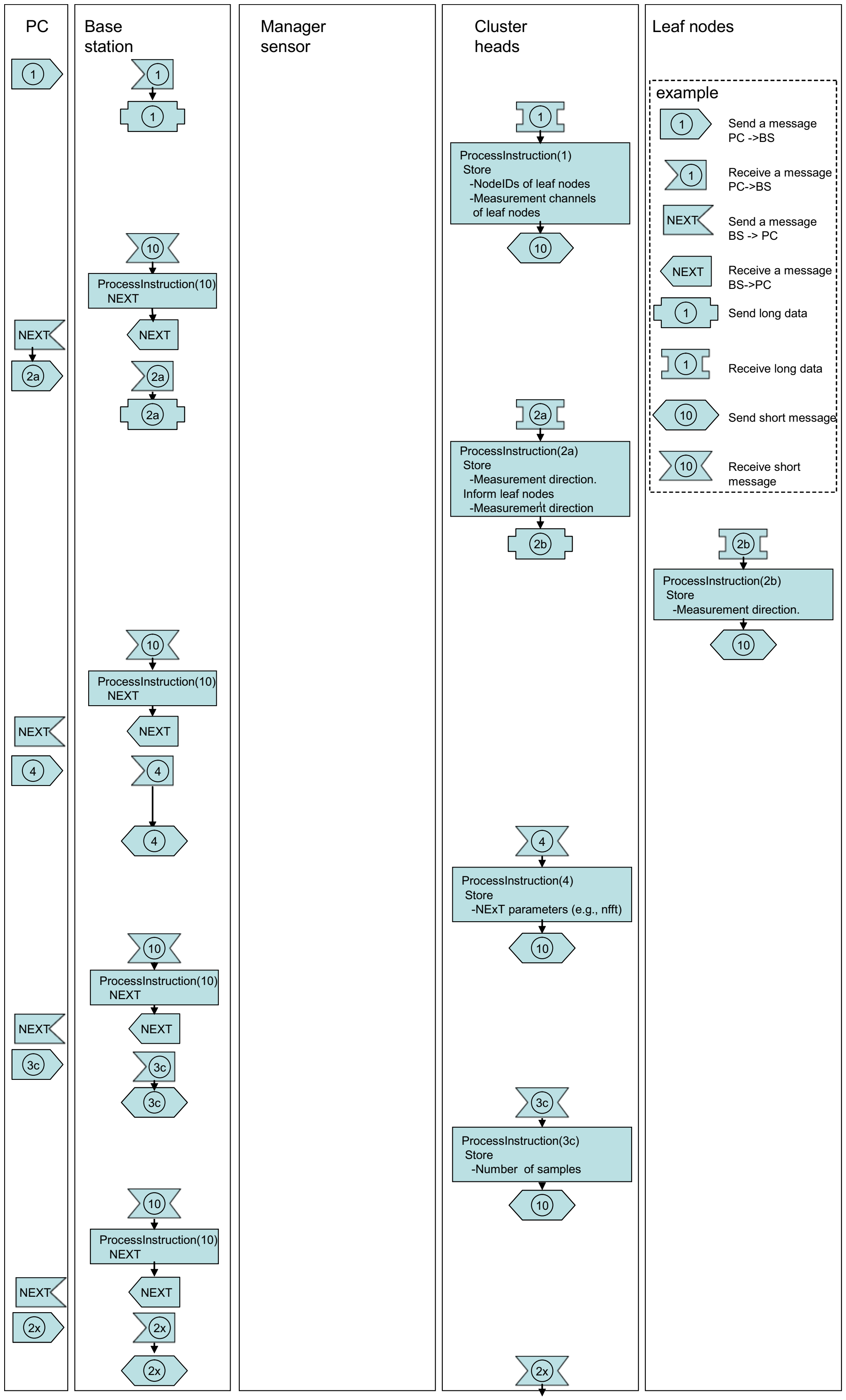

PC
Base station
Manager sensor
Cluster heads
Leaf nodes
example
Send a message PC ->BS
Receive a message PC->BS
Send a message BS -> PC
Receive a message BS->PC
Send long data
Receive long data
Send short message
Receive short message
ProcessInstruction(1) Store -NodeIDs of leaf nodes -Measurement channels of leaf nodes
ProcessInstruction(10) NEXT
ProcessInstruction(2a) Store -Measurement direction. Inform leaf nodes -Measurement direction
ProcessInstruction(2b) Store -Measurement direction.
ProcessInstruction(4) Store -NExT parameters (e.g., nfft)
ProcessInstruction(3c) Store -Number of samples
1
10
2a
2b
4
3c
2x
NEXT

Figure 7.26. The implementation block diagram of monitoring in a sensor community (1).

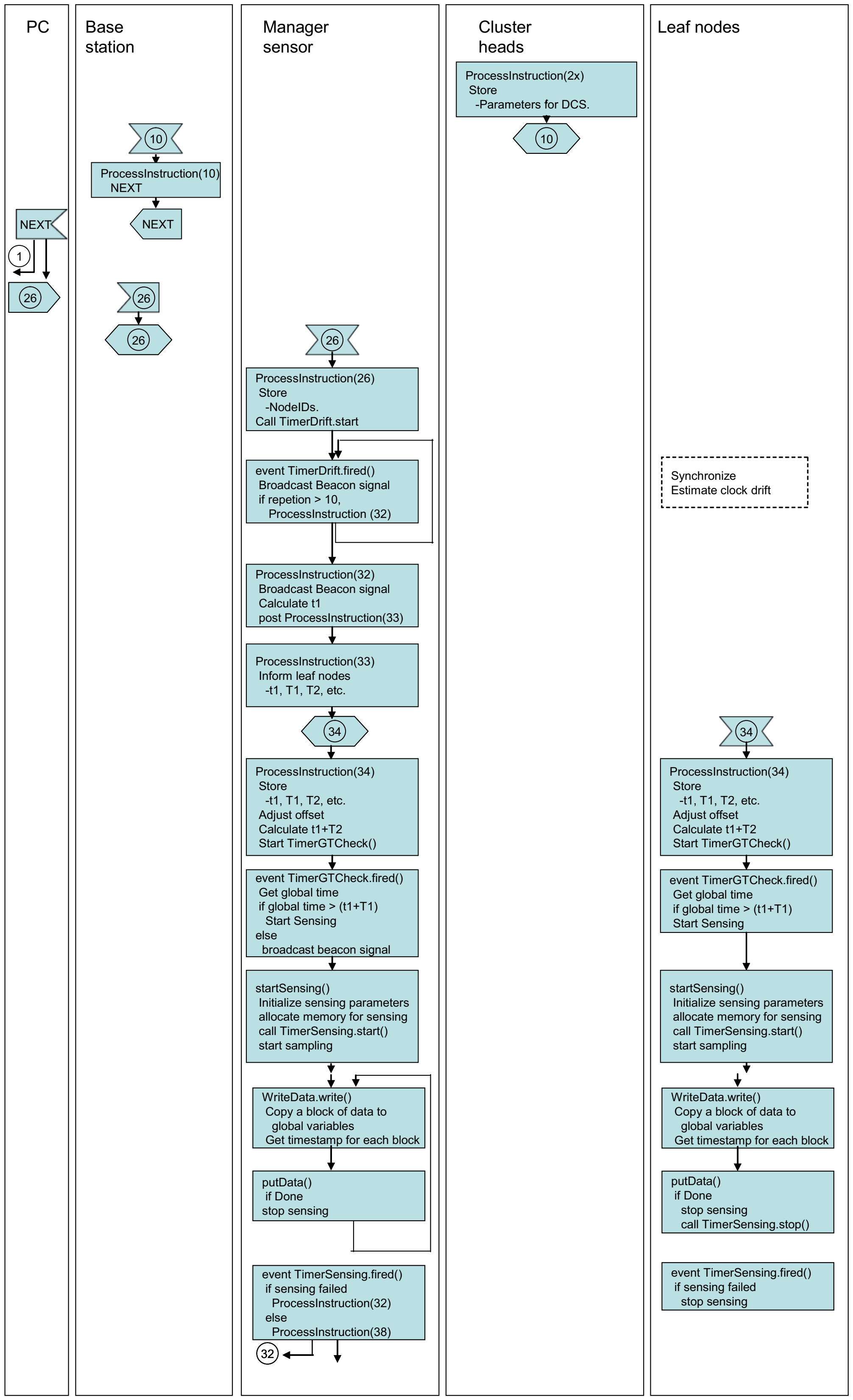

PC
Base station
Manager sensor
Cluster heads
Leaf nodes
ProcessInstruction(2x) Store -Parameters for DCS.
10
ProcessInstruction(10) NEXT
NEXT
NEXT
1
26
ProcessInstruction(26) Store -NodeIDs. Call TimerDrift.start
event TimerDrift.fired() Broadcast Beacon signal if repetion > 10, ProcessInstruction (32)
Synchronize Estimate clock drift
ProcessInstruction(32) Broadcast Beacon signal Calculate t1 post ProcessInstruction(33)
ProcessInstruction(33) Inform leaf nodes -t1, T1, T2, etc.
34
ProcessInstruction(34) Store -t1, T1, T2, etc. Adjust offset Calculate t1+T2 Start TimerGTCheck()
event TimerGTCheck.fired() Get global time if global time > (t1+T1) Start Sensing else broadcast beacon signal
event TimerGTCheck.fired() Get global time if global time > (t1+T1) Start Sensing
startSensing() Initialize sensing parameters allocate memory for sensing call TimerSensing.start() start sampling
WriteData.write() Copy a block of data to global variables Get timestamp for each block
putData() if Done stop sensing
putData() if Done stop sensing call TimerSensing.stop()
event TimerSensing.fired() if sensing failed ProcessInstruction(32) else ProcessInstruction(38)
event TimerSensing.fired() if sensing failed stop sensing
32

Figure 7.27. The implementation block diagram of monitoring in a sensor community (2).

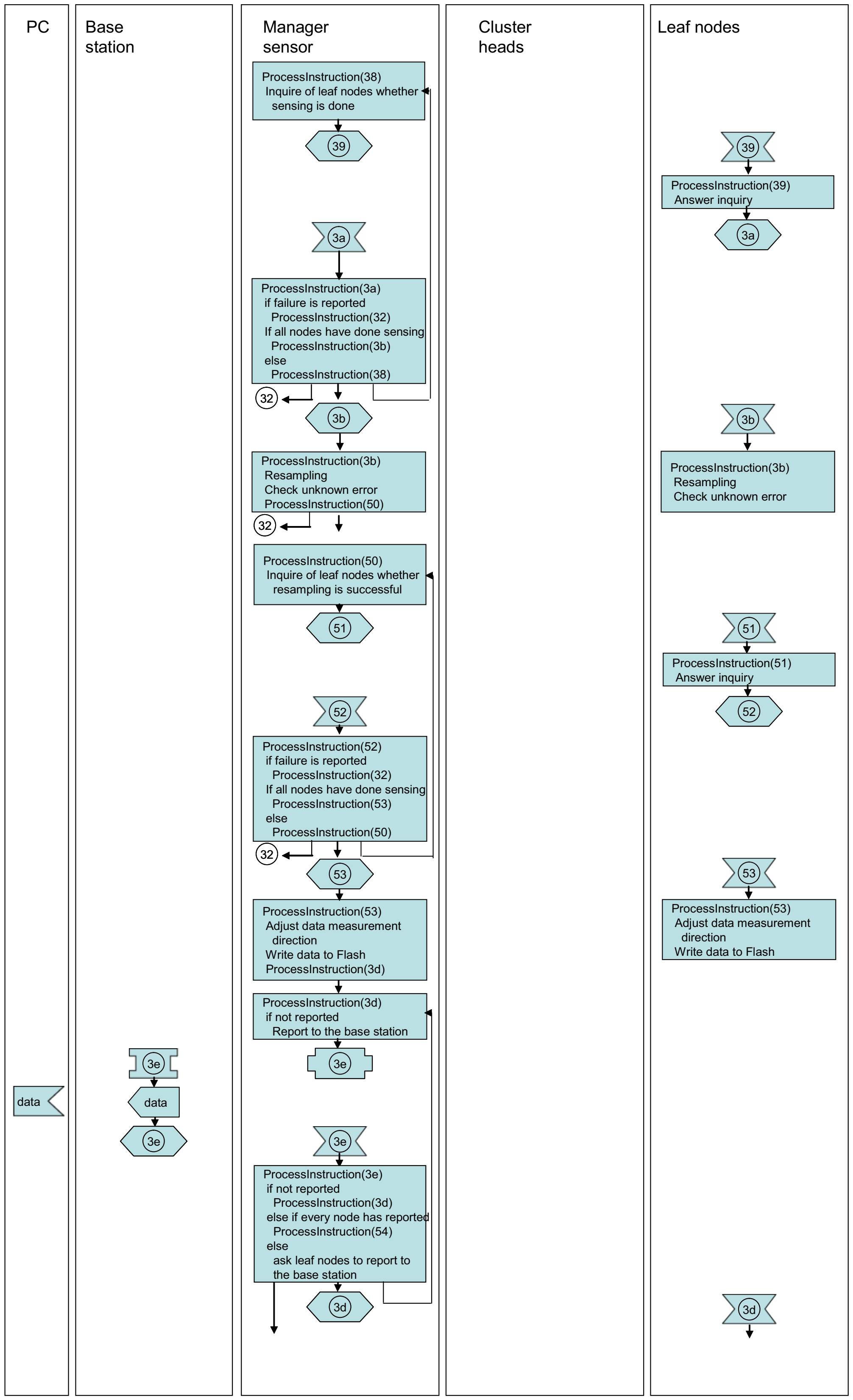


Figure 7.28. The implementation block diagram of monitoring in a sensor community (3).

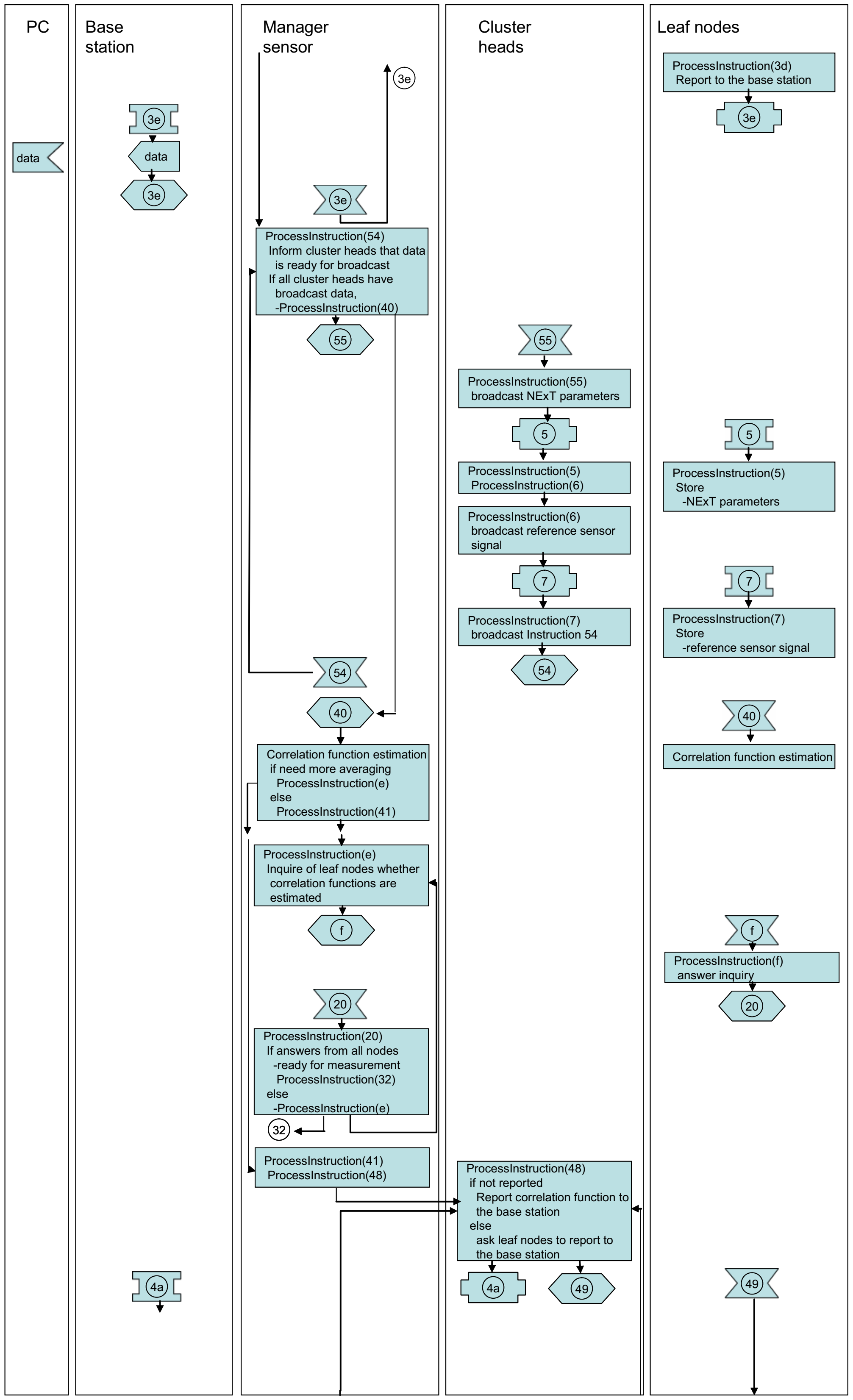

PC
Base station
Manager sensor
Cluster heads
Leaf nodes
3e
data
data
3e
3e
3e
ProcessInstruction(3d) Report to the base station
3e
ProcessInstruction(54) Inform cluster heads that data is ready for broadcast If all cluster heads have broadcast data, -ProcessInstruction(40)
55
55
ProcessInstruction(55) broadcast NExT parameters
5
5
ProcessInstruction(5) ProcessInstruction(6)
ProcessInstruction(5) Store -NExT parameters
ProcessInstruction(6) broadcast reference sensor signal
7
7
ProcessInstruction(7) broadcast Instruction 54
ProcessInstruction(7) Store -reference sensor signal
54
54
40
40
Correlation function estimation if need more averaging ProcessInstruction(e) else ProcessInstruction(41)
Correlation function estimation
ProcessInstruction(e) Inquire of leaf nodes whether correlation functions are estimated
f
f
ProcessInstruction(f) answer inquiry
20
20
ProcessInstruction(20) If answers from all nodes -ready for measurement ProcessInstruction(32) else -ProcessInstruction(e)
32
ProcessInstruction(41) ProcessInstruction(48)
ProcessInstruction(48) if not reported Report correlation function to the base station else ask leaf nodes to report to the base station
4a
4a
49
49

Figure 7.29. The implementation block diagram of monitoring in a sensor community (4).

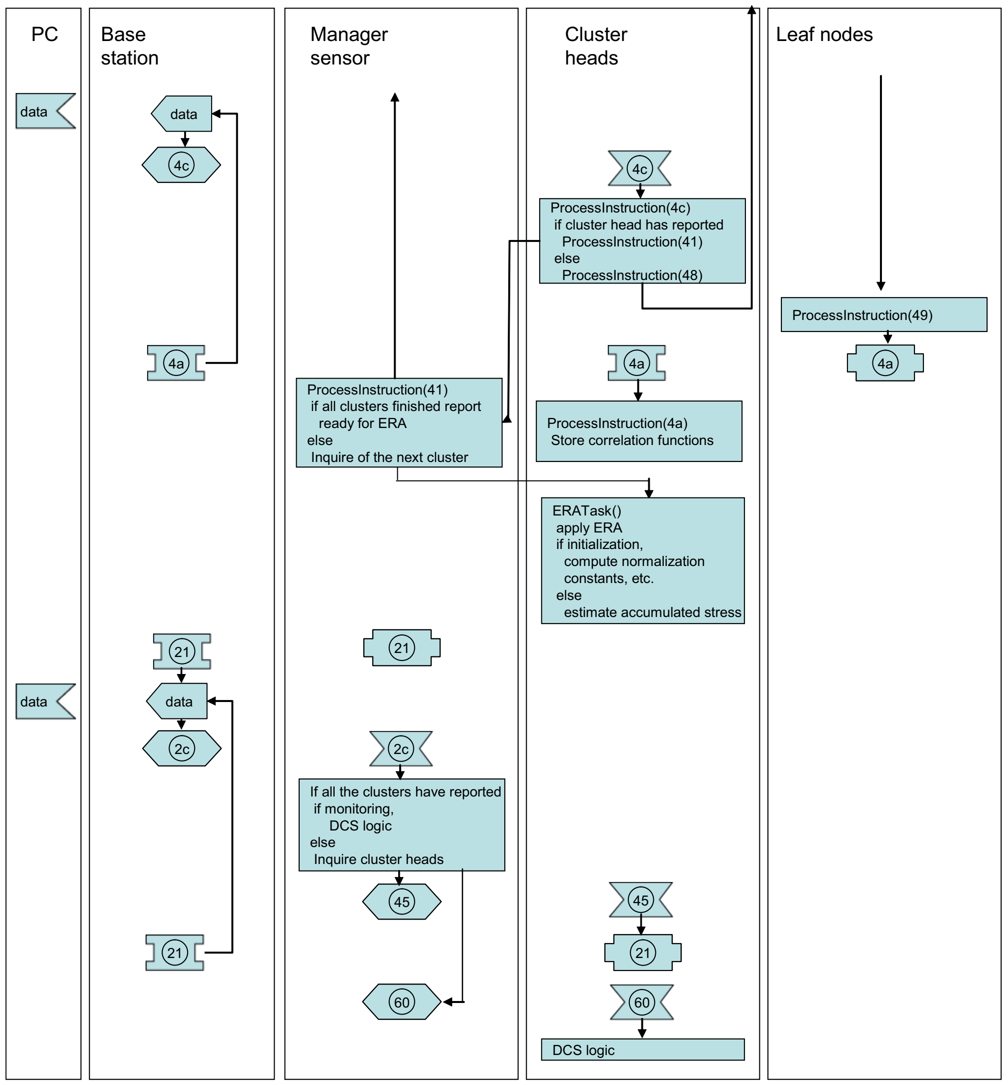


Figure 7.30. The implementation block diagram of monitoring in a sensor community (5).

Table 7.7. *Instructions in DCS Code*

Instruction	*Associated Operations*	*Accompanying Information*
1	The cluster heads save node IDs and measurement channels of their leaf nodes.	Node IDs of cluster members, measurement channels of each member node
10	The base station tells the PC that the base station is ready to receive the next instruction.	Acknowledgment
2a	Cluster heads save measurement directions of member nodes and forward this information to the leaf nodes.	Measurement directions of nodes
2b	The leaf nodes save measurement directions.	Measurement directions of cluster member nodes
4	The cluster heads save NExT parameters.	NExT parameters such as the number of FFT data points
3c	Sensing parameters are saved on cluster heads.	Sensing parameters such as Number of samples
22-25	Received parameters are saved on cluster heads.	The number of elements, weight of additional mass, results from the previous set of measurement, etc.
26	The manager receives node IDs of all the nodes and cluster heads. The manager then starts time synchronization and estimates clock drift.	Node IDs of all the leaf nodes and cluster heads
32	The manager broadcasts beacon signal for time synchronization.	
33	The manager notifies leaf nodes of timings, t1, T1, and T2.	
34	All of the nodes save t1, T1, and T2. A timer is called to periodically check timing to start sensing.	Timing to start sensing, i.e. t1, T1, T2, etc.
39	The receivers reply to inquiry message about sensing completion.	
3a	The manager judges whether to keep inquiry, restart sensing, or apply resampling.	Answer to inquiry about sensing completion
3b	All of the sensors start resampling, check for unknown error. The manager then posts ProcessInstruction(50).	
50	The manager asks the leaf nodes whether the resampling was successful.	
51	The leaf nodes reply to the inquiry about successful resampling.	
52	The manager judges whether to keep inquiry, restart sensing, or adjust measurement directions.	Answer to inquiry about successful resampling

Table 7.7. *Instructions in DCS Code (Continued)*

Instruction	*Associated Operations*	*Accompanying Information*
53	All of the nodes adjust measurement data for sensor direction. Nodes then write data to the Flash memory.	Sensing was successfully completed at all of the nodes.
3d	If measurement data has not been reported to the base station, the node reports the data to the base station.	
3e	If the receiver has not reported its data, the receiver reports to the base station. If all of the leaf nodes have not reported their data, the manger asks them to report. If all of the nodes have finished reporting, the manager asks cluster heads to broadcast data to their members.	Raw data/acknowledgment from the base station
55	Cluster heads broadcast NExT parameters to their members.	
5	The sender prepares to broadcast the reference signal. The receiver stores NExT parameters.	The number of FFT data points, number of averages, etc.
6	The cluster heads broadcast reference signals.	
7	The sender reports to the manager that the broadcast was successful. The receiver stores reference sensor signal.	Reference sensor signal
54		Report to the manager about successful broadcast.
40	All of the nodes start correlation function estimation.	
e	The manager inquires all of the leaf nodes whether correlation function estimation is complete.	
f	The receiver replies to the inquiry.	Inquiry about correlation function estimation completion
20	If all of the nodes have replied, the manager starts sensing process by posting ProcessInstruction(32). If not, keep inquiry.	Answer to the inquiry
48	If correlation function estimate on a cluster head is not reported to the base station, the cluster head report to the base station. Otherwise, the cluster head asks leaf nodes to report their correlation functions to the cluster head and base station.	
49	The receiver reports the correlation function estimates to the cluster head and base station.	Inquiry about correlation function estimate
4a		Correlation function estimates

Table 7.7. *Instructions in DCS Code (Continued)*

Instruction	*Associated Operations*	*Accompanying Information*
4c	If the cluster head has collected all of the correlation functions from its members, report to the manager. Otherwise, the cluster head keeps asking leaf nodes for their correlation function estimates.	
41	If all cluster heads finished collecting correlation function estimates from their members, the manger asks cluster heads to perform ERA. Otherwise, ask the next cluster head to collect correlation function estimates.	
21		ERA results
2c	If all of the clusters have not reported ERA results, the manager asks cluster heads to report. After all of the clusters report to the base station, DCS logic is initiated if the system is monitoring structures to detect damage.	Acknowledgment
45	The manager asks cluster heads to report ERA results to the base station.	
60	The receiver initiates DCS logic.	

Chapter 8

EXPERIMENTAL VERIFICATION

This chapter provides experimental verification of the SHM framework developed in this research. The SHM framework employing the DCS for SHM was realized on networks of Imote2s in Chapter 7. Numerical operations of the SHM system were numerically examined in a component-by-component manner by injecting acceleration response data to the network of Imote2s and processing the data. The system has been shown to be capable of processing acceleration data and localizing damage.

The experimental verification of the proposed SHM framework uses acceleration response data measured by networks of Imote2s. Acceleration measurements and the subsequent data processing are performed before and after an element of the truss is replaced with a thinner element having a 52.7 percent cross-section reduction. The replaced element is detected by cluster head Imote2s. The experiment is repeated multiple times to assess reproducibility. Also, the experiment is repeated by replacing a different structural element to see the damage detection capability of the system in various elements. In this way, the ability of the proposed framework to localize damage is experimentally verified (Spencer & Nagayama, 2006).

8.1 Experimental setup

The SHM framework is experimentally verified on networks of Imote2s using the 5.6m-long, three-dimensional truss structure discussed in Chapter 7. The truss and the structural node IDs and element IDs are shown again in Figures 8.1 and 8.2. The shaker is attached at node 17 and excites the truss with a band-limited white noise.

The Imote2s are installed on the truss using plastic fixtures (see Figure 8.3). The fixture has two walls, each of which has two threaded holes. A set-screw is inserted into

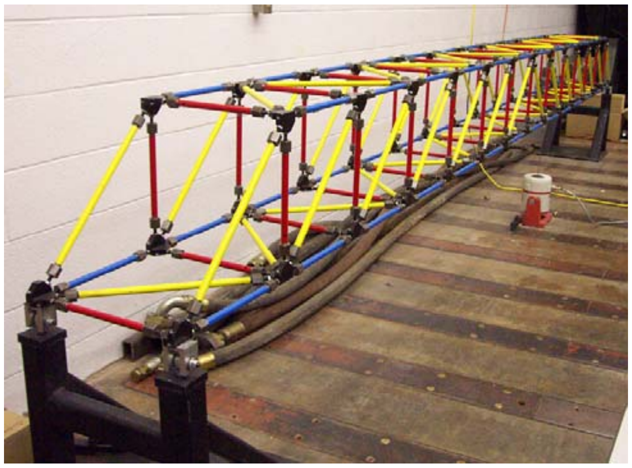


Figure 8.1. The Imote2 and plastic case.

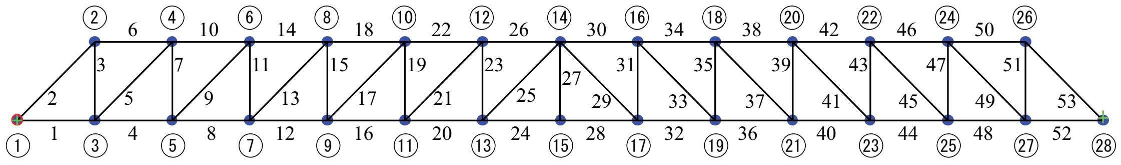


Figure 8.2. Pin and roller ends.

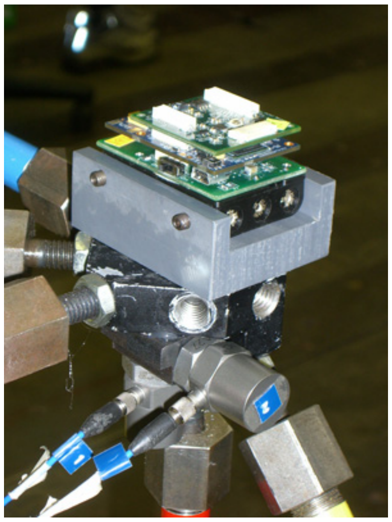


Figure 8.3. Plastic fixture and the Imote2.

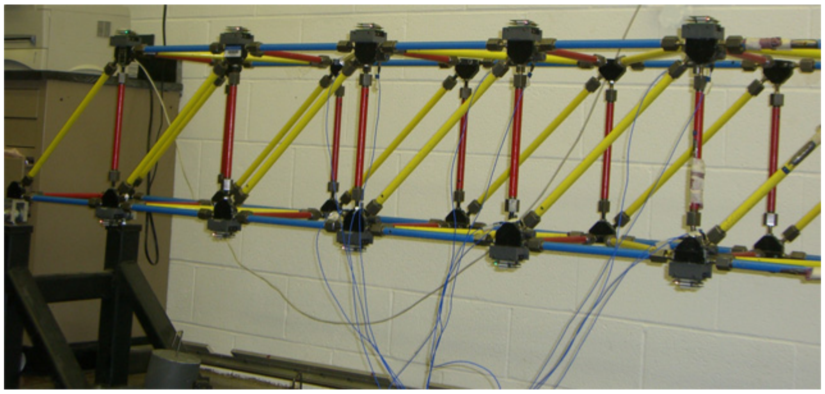


Figure 8.4. Ten Imote2s installed on nodes 2 to 11.

each of the threaded holes and tightened. The Imote2 is, thus, supported by the bottom of the plastic fixture and the four set-screws.

Ten Imote2s are placed on the structure on the front panel of the truss creating three overlapping local sensor communities. Because the number of available Imote2 sensors is limited, sensors are moved on the structure, depending on which element is considered as damaged. When damage in element 8 is considered, Imote2s are installed on nodes 2 to 11, with nodes 4, 6, and 8 becoming cluster heads that organize the local sensor communities (see Figure 8.4). For example, node 4 organizes the local sensor community consisting of nodes 2, 3, 4, 5, 6, and 7. When damage in element 20 is considered, Imote2s

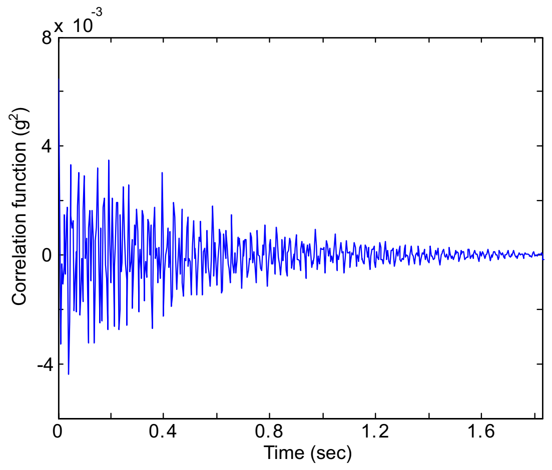


Figure 8.5. Correlation function estimate from Imote2 data.

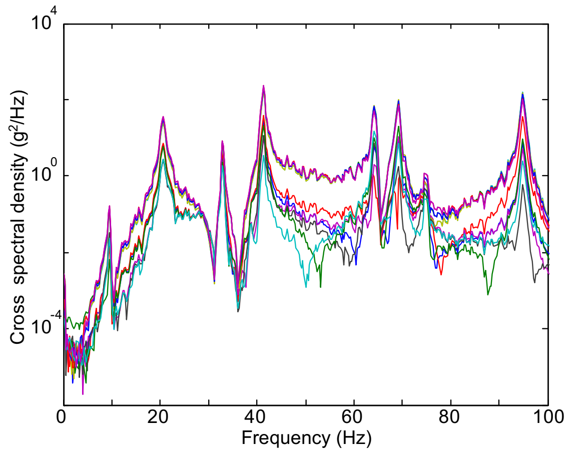


Figure 8.6. Cross-spectral density estimates from Imote2 data.

are placed on nodes 6 to 15, and nodes 8, 10, and 12 become cluster heads. Local sensor communities organized by nodes 4, 6, 8, 10, and 12 are denoted as sensor communities 1, 2, 3, 4, and 5, respectively. Each node measures acceleration in three directions. Because this validation exercise is monitoring the vertical plane of the truss, only longitudinal and vertical acceleration records are utilized.

8.2 NExT

The acceleration responses are measured by Imote2s, and the correlation functions are estimated. The vertical acceleration signal at the cluster head node is used as the reference signal. One of the estimated correlation functions and the associated frequency domain representation, the cross-spectral densities, are shown in Figures 8.5 and 8.6. These estimates show agreement with those estimated on a PC using the same data from Imote2s.

8.3 ERA

ERA results during monitoring are summarized in Table 8.1. ERA on a PC using the same data gives the same results numerically. Note that the structure has undergone some structural changes since the data injected to Imote2 networks in Chapter 7 was acquired; some of connections were tightened and some of elements were adjusted. The identified natural frequencies in Table 8.1 are not necessarily same as those in Table 7.3. By comparing the identified natural frequencies with the cross-spectral density plots in Figure 8.6, the identified natural frequencies are considered to well capture the dynamic characteristics of the signals.

Table 8.1. *Identification of Natural Frequencies and Damping Ratios*

Natural Frequency (Hz)	*Damping Ratio (%)*
20.7229	2.9321
32.8985	0.2127
41.2990	0.4846
64.3174	0.2422
69.1086	0.3718
94.9259	0.2810

8.4 DLV methods

The DLV method based on the mass perturbation and SDLV methods are applied to the modal parameters identified before and after damage. The mass perturbation DLV method, involving initialization to determine the mass normalization constants is first examined. Subsequently, the SDLV method is investigated.

The mass normalization constants to be used in the mass perturbation DLV method are first estimated. Imote2s are installed at nodes 7, 9, 11, 13, 15, 17, 19, 21, 23, and 25. Measurements are taken before and after additional mass is attached to node 11. The mass is 1.253 kg. All the nodes measure vertical acceleration. The node at the location of the additional mass measures acceleration in the longitudinal and transverse directions as well, because the denominator of Eq. (6.21) requires mode shapes at the degrees-of-freedom corresponding to the mass perturbation, i.e., acceleration in all three directions is needed. Thus, acceleration is measured in 12 channels. The mass normalization constants are estimated and listed in Table 8.2, as well as natural frequencies of the structure with and without the mass perturbation.

The identified mass normalization constants are utilized to reconstruct the flexibility matrix as shown in Eqs. (6.19) and (6.32). Imote2s monitoring sensor communities 1, 2, and 3 reconstruct the flexibility matrices before and after element 8 is replaced with an element of reduced cross-section. The Imote2s then perform an SVD on the difference in the flexibility matrices to estimate the DLVs. The DLVs are then multiplied by a matrix to convert nodal force to element stresses. Elements with small stresses are identified as

Table 8.2. *Mass Normalization Constant Estimates from Imote2 Data*

Natural Frequency (Hz)		*Mass Normalization Constant*
Without Mass Perturbation	*With Mass Perturbation*	
20.4680	20.3722	0.004776
32.7488	32.1563	0.001474
41.2410	40.9012	0.007941
64.5915	64.0984	0.009026
69.3455	69.0262	0.005865
95.5995	95.1737	0.011405

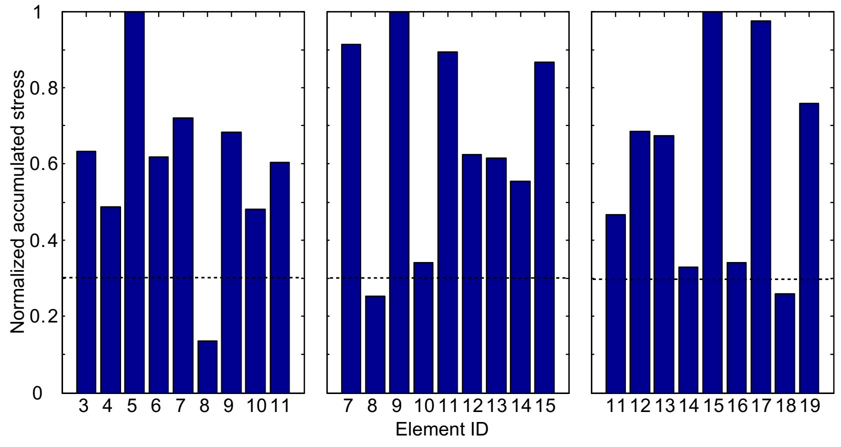


Figure 8.7. Normalized accumulated stress in sensor community 1, 2, and 3 (the mass perturbation DLV method).

potentially damaged elements. Figure 8.7 shows the normalized accumulated stress calculated for the three sensor communities. Element 8 is identified as the element with a normalized accumulated stress smaller than the threshold value of 0.3. Note that the stress in element 18 is also small. The DLV method identifies potential damage sites as elements with small stress. The method does not exclude undamaged elements from a set of elements with small stress. Gao (2005) pointed out that such false-positives in damage detection do not appear repeatedly; repetitive measurements and damage detection can distinguish damaged elements as elements which consistently show small stress. The results in Figure 8.7 support that the mass perturbation DLV method on Imote2s can detect damage.

The SDLV method is then employed as the damage detection method. This method does not require estimation of mass normalization constants illustrated in Figure 7.24. The same set of sensor communities, 1, 2, and 3 again measures acceleration responses before and after element 8 is replaced with an element of reduced cross section; subsequently, the SDLV method is applied. Using the conversion matrix, the normalized accumulated stresses are calculated from DLVs. Figure 8.8 shows the accumulated stresses estimated

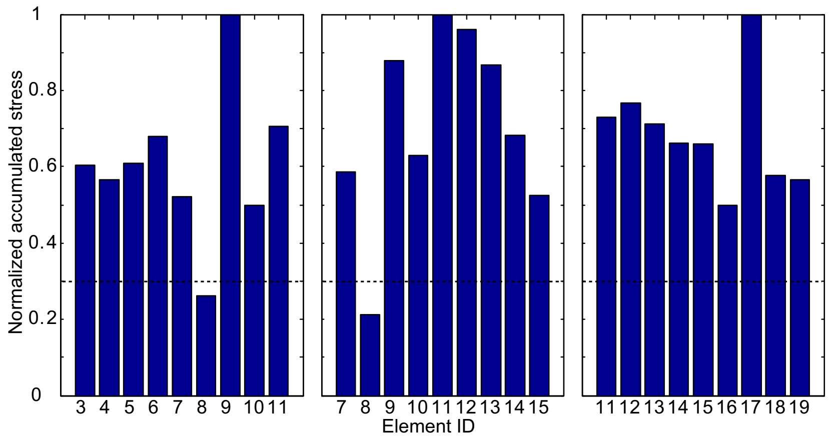


Figure 8.8. Normalized accumulated stress in sensor community 1, 2, and 3 (the SDLV method).

by the three cluster head nodes. Element 8 is detected as an element with the normalized accumulated stress smaller than the threshold value, 0.3.

The intermediate and final results of the two DLV methods such as singular values, DLVs, and the normalized accumulated stress are also calculated on a PC based on the centrally collected acceleration time histories. The results calculated on the Imote2 are the same as those on the PC with the same precision of the data type.

The damage detection capability of the proposed SHM framework implemented on a network of Imote2s has been demonstrated in this section. Further investigation of the capabilities of the approach will be pursued in section 8.8 by replacing other elements of the truss.

8.5 DCS logic

Upon completing calculation of the normalized accumulated stresses, the three cluster heads exchange their damage detection results to perform the DCS logic given in Figure 7.22. The log of the associated session recorded on the base station is presented in Figures 8.9 and 8.10. These figures correspond to the damage detection results in Figures 8.7 and 8.8, respectively. Because no inconsistency is reported, retake flags are set to zero as indicated by lines2, 9, and 16 in Figures 8.9 and 8.10. When no inconsistency is found, the results are reported to the base station, and the Imote2s prepare to enter a sleep mode. Though element 18 is identified as a damaged element by the mass perturbation DLV method, this false-positive is likely to be avoided if sensor community 4 also monitors this element because of the redundancy. When inconsistency is detected, the DCS algorithm is repeatedly applied. Figure 8.11 shows the report to the base station when inconsistency is observed. Sensor communities 3, 4, and 5 monitor the truss to detect damage at element 21. Sensor communities 3 and 4 organized by cluster head nodes 73 and 135 find inconsistency. Sensor community 3 identifies element 16 as damaged, while sensor community 4 does not. After all three cluster heads finish reporting, only sensor communities 3 and 4 repeat acceleration measurement and data processing. The DCS

```
 1
 2 Node ID 67 RETAKE 0
 3
 4 ID 67 # of Damaged Element: 1
 5
 6 Damaged Element ID: 8
 7
 8
 9 Node ID 81 RETAKE 0
10
11 ID 81 # of Damaged Element: 1
12
13 Damaged Element ID: 8
14
15
16 Node ID 73 RETAKE 0
17
18 ID 73 # of Damaged Element: 1
19
20 Damaged Element ID: 18
21
```

Figure 8.9. DCS logic log (the mass perturbation DLV method).

```
 1
 2 Node ID 67 RETAKE 0
 3
 4 ID 67 # of Damaged Element: 1
 5
 6 Damaged Element ID: 8
 7
 8
 9 Node ID 81 RETAKE 0
10
11 ID 81 # of Damaged Element: 1
12
13 Damaged Element ID: 8
14
15
16 Node ID 73 RETAKE 0
17
18 ID 73 # of Damaged Element: 0
```

Figure 8.10. DCS logic log (the SDLV method).

logic is applied again, and the results are reported to the base station as shown from lines 30 to 46 in Figure 8.11. No inconsistency is reported this time and all of the Imote2s are ready to enter the sleep mode.

```

Node ID 73 RETAKE 1
INCONSISTENT

Inconsistent Element ID: 16

ID 73 # of Damaged Element: 1

Damaged Element ID: 16


Node ID 135 RETAKE 1
INCONSISTENT

Inconsistent Element ID: 16

ID 135 # of Damaged Element: 1

Damaged Element ID: 21


Node ID 70 RETAKE 0

ID 70 # of Damaged Element: 1

Damaged Element ID: 21

....

Node ID 73 RETAKE 0

ID 73 # of Damaged Element: 0


Node ID 135 RETAKE 0

ID 135 # of Damaged Element: 1

Damaged Element ID: 21


Node ID 70 RETAKE 0

ID 70 # of Damaged Element: 1

Damaged Element ID: 21

```

Figure 8.11. DCS logic log with inconsistency (the SDLV method).

8.6 Calculation and communication time

The Imote2 communication and calculation speeds are limited, raising concerns over the time necessary to perform the SHM strategy. The shorter time is preferable from two perspectives: power consumption and monitoring interval. Although longer time required for a SHM strategy does not necessarily imply larger power consumption, the time can be a rough indicator of power consumption. If nodes do not enter a sleep mode or turn off radio components during monitoring, their power consumption correlates well with the time of operation. With respect to monotoring interval, its frequency is limited by the length of execution time; an SHM strategy that takes four hours to complete, for example, can be performed at most only once in four hours. Considering monitoring of structures on a weekly or monthly basis, Imote2s do not need to complete the SHM strategy in a short time, e.g. one or two hours. However, monitoring in a short time is preferable to allow multiple measurements before environmental conditions such as wind velocity and temperature change. When damage detection results from neighboring sensor communities are inconsistent with each other, sensing and damage detection need to be repeated. If the SHM strategy takes a long time, environmental conditions may change between monitoring intervals, making comparison among measurements difficult. Therefore, SHM strategies requiring a short execution time are preferable.

In this section, the time required for execution of the DCS for SHM on Imote2s is determined. The implementation includes numerous reports sent to the base station for debugging purposes. Considering that these tasks can be omitted in the future, the time necessary for the SHM strategy is estimated.

The breakdown of the time spent during DCS for SHM is listed in Table 8.3. At the beginning of monitoring, parameter initialization requires about 70 seconds. Parameters to be initialized include: the conversion matrix from nodal force to stress in elements, modal parameters identified before damage, sampling rate, and duration of sensing. A six-second-long time synchronization activity follows the parameter initialization. Sensing continues for approximately 50 seconds. Note that the duration of sensing can be increased if sufficient RAM is available for temporary storage of the measured data. At the end of sensing, the manager sensor queries each node to see if sensing was successful. Then the resampling process is applied to the measured data; resampling takes ten seconds. The manager node again queries each node to ensure successful resampling. The measured data is then reported to be the base station. The communication between a leaf node and the base station is about five seconds, while communication between the base station and a PC takes 12 seconds. Each Imote2 node reports three axes of measurement data. This reporting is one of the tasks requiring significant time. Then the cluster heads multicast their own data to the leaf nodes as reference signals. NExT parameters such as the number of averages and the number of FFT points are also multicast. This multicast of parameters and data takes only 33 seconds for three sensor communities. Then NExT is applied on each node. Nodes which belong to multiple sensor communities estimate multiple sets of correlation functions. The estimated correlation functions are sent to the cluster heads and the base station. The base station then forwards the correlation functions to the PC. This correlation function report takes more than 400 seconds. Finally, ERA is applied to the correlation functions, and the DLV method determines the potential locations of damaged elements. This data processing on the cluster heads takes 150 seconds. In total, DCS for SHM takes more than 23 minutes.

Among the tasks listed in Table 8.3, some of them can be omitted when debugging is not necessary. The raw data does not need to be sent to the base station. About 500

Table 8.3. *Breakdown of the Time Spent During DCS for SHM*

Task	*Time (sec) With Debugging Purpose Tasks*	*Time (sec) Without Debugging Purpose Tasks*
1. Parameter injection to cluster heads and the manager node	*70*	*0*
2. Time synchronization	*6*	*6*
3. Sensing	*50*	*50*
4. Query	*10*	*10*
5. Resampling	*10*	*10*
6. Query	*5*	*5*
7. Report to the base station	*5*	*0*
8. Communication between the base station and a PC	*12*	*0*
9. Repeat 7 and 8 (# of nodes x *3 axis -1) times.*	*~ 493 (=17*x*29)*	*0*
10. Multicast the NExT parameters.	*5*	*5*
11. Multicast reference signals.	*6*	*6*
12. Repeat 10 and 11 (# of cluster head nodes-1) times.	*~ 22 (=11*x*2)*	*22*
13. NExT	*42*	*42*
14. Repeat 13 (# of sensor communities a node belongs to) times.	*~84 (=42*x*2)*	*84*
15. Report correlation functions to cluster heads and the base station.	*6*	*6*
16. Communication between the base station and a PC	*6*	*0*
17. Repeat 15 and 16 (# of cluster heads x *# of leaf nodes in a community* x *2 - 1) times.*	*~420 (= 35*x*12)*	*174*
18. ERA/DLV	*150*	*150*
Total	*1,402*	*570*

seconds spent for this reporting are saved. Also, correlation function reports do not need to be sent to the base station and the PC. Six seconds spent for the communication in task 16 are saved. The 420 seconds listed at task 17 can be reduced by a factor of two. Furthermore, only the beginning portion of the correlation function is utilized to construct the Hankel matrix in Eq. (6.8), while the current system reports the entire correlation function to the base station and cluster head nodes. Therefore, the report to the cluster heads can be even shorter, though its contribution toward shortening the SHM execution is not counted here. Subtracting the time spent for these debugging tasks from the 1,402 seconds makes the total time for the SHM strategy approximately 676 seconds; the usage of 400 MHz mode of the CPU may further shorten the time necessary to execute the SHM strategy.

In summary, about one minute of sensing and 9 minutes of post processing is necessary for the current SHM system without sending debugging information to the base station. Note that the time needed for sensing may increase if natural frequencies of a structure appear in a lower frequency range. As for the time for post processing, the time does not increase as the number of sensor community increases. Though tasks 12, 14 and 17 in Table 8.3 seem to be proportional to the number of sensor communities, they are proportional only when the number of communities is small. When one cluster is out of the communication range of another cluster, these two clusters can perform the tasks in 12, 14, and 17 simultaneously. Monitoring damage on a large structure requiring only about nine minutes of data processing in addition to sensing time is an appealing feature of this approach.

8.7 Battery life

The battery life of the Imote2 while running the proposed SHM strategy implementation is estimated in this section. When Imote2s performed sensing, the change in the supply voltage to the Imote2s is recorded. In contrast to the previous section, sensing was repeated four times in this experiment. That is, tasks 2 to 14 in Table 8.3 were repeated four times. The voltage values of five Imote2s are read after parameters are initialized, after the first sensing is performed, after reporting to the base station is completed, after the second, third, and fourth measurement finishes, and at the end of ERA and DLV method applications.

Figure 8.12 shows the change in the supply voltage to the Imote2s. One round of measurement consumes about 0.03 to 0.08 V. Note that sometimes sensing fails and is repeated until successful measurement is performed. When repetition takes place many times, the decrease in the voltage is considered significant. If sensing becomes more stable with less sensing failures, the number of repetitions will decrease reducing the power consumption per round of sensing.

While Imote2s repeat sensing four times for the data shown in Figure 8.12, monitoring of full-scale structures may not need repeated measurements, thus reducing the power consumption per monitoring event. The advantage of repeating sensing is to reduce the effect of observation noise by increasing the number of averages in the spectral densities in Eq (5.1). The SHM implementation explained in sections 8.2 to 8.5, as well as that in section 8.8, measures acceleration responses of the truss only once, yielding 21 averages. From the damage detection results shown in section 8.4, 21 averages seem to be

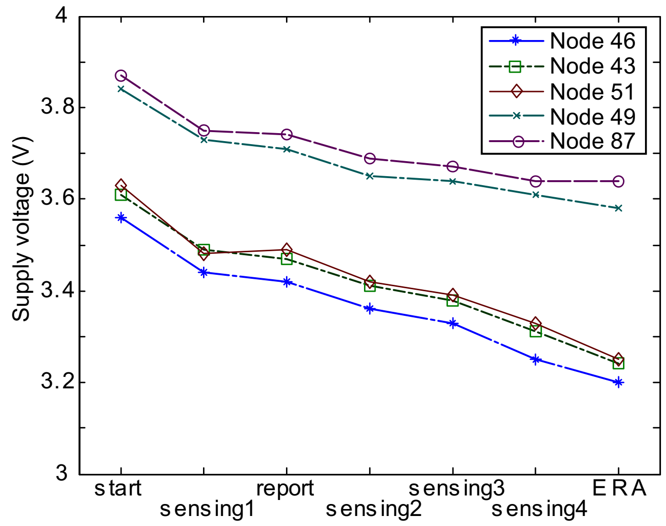


Figure 8.12. Change in the supply voltage to the Imote2.

sufficient. Although the change in supply voltage during these experiments without repeated measurements was not recorded, the battery life is estimated by counting the number of experiments conducted using one set of batteries. The Imote2 utilizes three AAA batteries. A new set of batteries supplies 4.5 to 4.8 V to the Imote2. The Imote2 can operate when the supply voltage is above 3.2 V. More than 30 sets of experiments were performed without changing batteries. Simply dividing the change in the supply voltage by 30 yields about a 0.05 V decrease per experiment when sensing is not repeated. If monitoring is performed on a weekly basis, a battery life providing 30 sets of measurements and damage detection calculations allows the Imote2 to monitor a structure for about eight months. Note that the Imote2 consumes power even in a sleep mode, shortening the battery life.

For monitoring of full-scale structures, sensing may continue longer, thereby consuming more power. For example, 40 seconds of acceleration data sampled at 280 Hz is utilized in the experiment and damage is successfully detected. However, a full-scale structure may have lower natural frequencies than the lowest one of the truss (i.e., the mode at about 20 Hz), necessitating longer acceleration measurements. Though a lower sampling frequency may be allowed for such structures, the duration of sensing is likely to be significantly longer than 40 seconds. Battery depletion may become greater than 0.05 V per monitoring event due to the longer sensing time.

8.8 Damage detection results

The damage detection capability of the SHM algorithms implemented on the Imote2 is examined in this section by monitoring the truss with its various elements replaced with a “damaged” element, i.e., one of reduced cross section. Elements 8, 9, 10, 11, 12, 19, 20, 21, and 22 are replaced one-by-one with the damaged element. For the first five cases, sensor communities 1, 2, and 3 participate in damage detection, while sensor communities

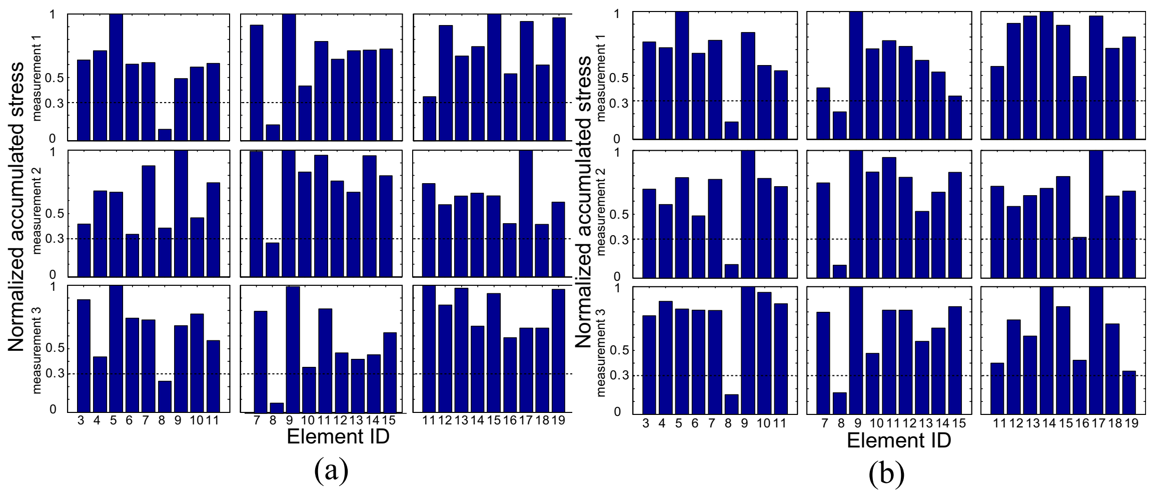


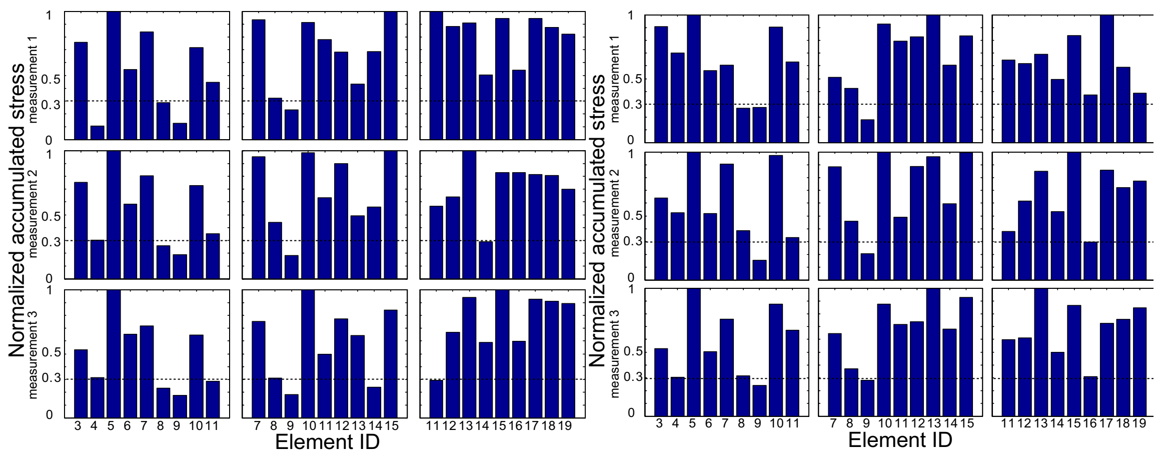
Figure 8.13. Damage detection (damaged element: Element 8): (a) mass perturbation DLV; and (b) SDLV.

3, 4, and 5 monitor for the damaged element for the other cases. The experiment is repeated at least three times for each case to assess the repeatability of damage detection.

The SDLV method is employed as the DLV method on Imote2s, while the damage detection capability of the mass perturbation DLV method is also examined on a PC. As stated in section 8.4, the DCS for SHM using known mass perturbation has been implemented on Imote2s, and the same numerical operations on Matlab on the centrally collected truss acceleration response data has been confirmed to produce the same calculation results as those on Imote2s. Therefore, for convenience, the mass perturbation DLV method is applied on Matlab to the centrally collected data to investigate the damage detection capability.

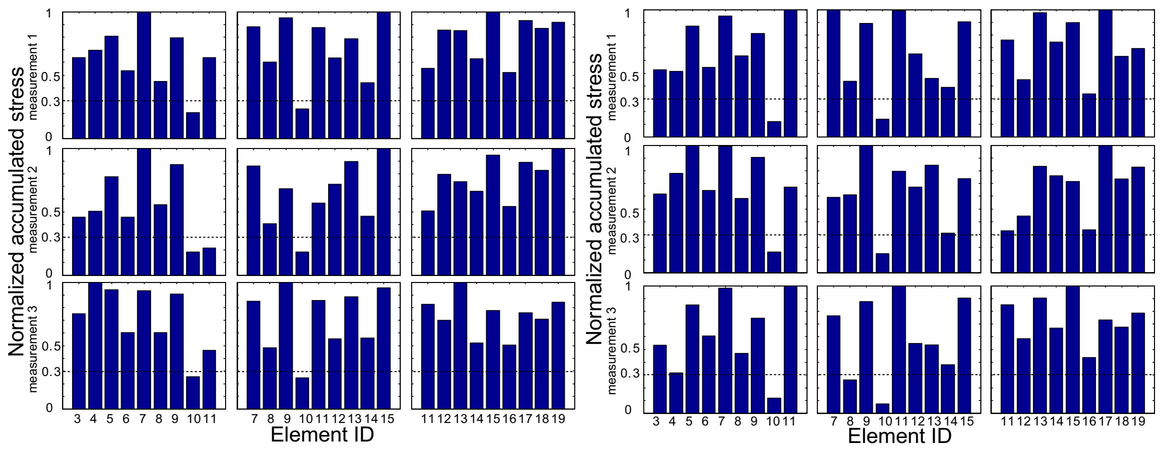
Figures 8.13 to 8.21 show the damage detection results using the two DLV methods. For most of the experiments, both of the DLV methods can detect the damaged element, i.e., it has a normalized accumulated stress smaller than 0.3. In some cases, however, false-positive and/or false-negative damage detection is observed.

False-positive damage detection can be theoretically explained. The DLV methods identify elements with small stress as candidates for damaged elements. Undamaged elements are not necessarily excluded from the small stress elements. Bernal (2002) and Gao (2005) reported that such false-positives are occasional in their simulations and experiments. In the Imote2 experiments, Figure 8.15 (a) shows false-positive damage detection at element 11 in the second measurement. This false-positive is not observed in the first and third measurements. In Figure 8.14 (a), however, false-positives are observed consecutively. Element 8 has small stress in all cases. Though consecutive false-positives do not contradict the theoretical derivation of the DLV methods, they are not reported in Bernal (2002) or Gao (2005). Consecutive false-positives are likely to indicate that factors such as a structure's nonlinearity and observation noise affect the damage detection capability in these experiments.



(a) (b)

Figure 8.14. Damage detection (damaged element: Element 9): (a) mass perturbation DLV; and (b) SDLV.



(a) (b)

Figure 8.15. Damage detection (damaged element: Element 10): (a) mass perturbation DLV; and (b) SDLV.

From a theoretical perspective, false-negatives, on the other hand, are not expected if conditions such as linearity of the structure and no observation noise are satisfied. No false-negative damage detection is found in Bernal (2002) and Gao (2005). In the Imote2 experiments, false-negatives are observed. For example, Figure 8.16 (a) shows the damage detection results when element 11 is replaced with the damaged element. Seven false-negative damage detection cases are observed in this figure. The SHM system cannot detect damage in this case, even though the three sensor communities commonly monitor the replaced element. Gao (2005) stated that detecting damage in the vertical and diagonal elements located close to the midspan of the truss structure was found to be more difficult than for other elements. Damage detection of elements carrying small force under self-weight is considered challenging. Small forces in vertical element 11 under self-

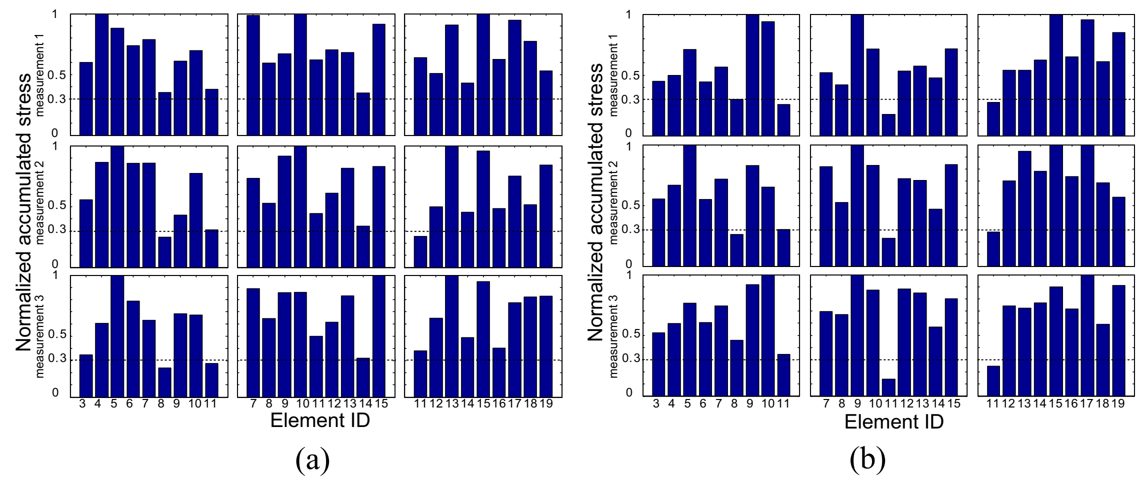


Figure 8.16. Damage detection (damaged element: Element 11): (a) mass perturbation DLV; and (b) SDLV.

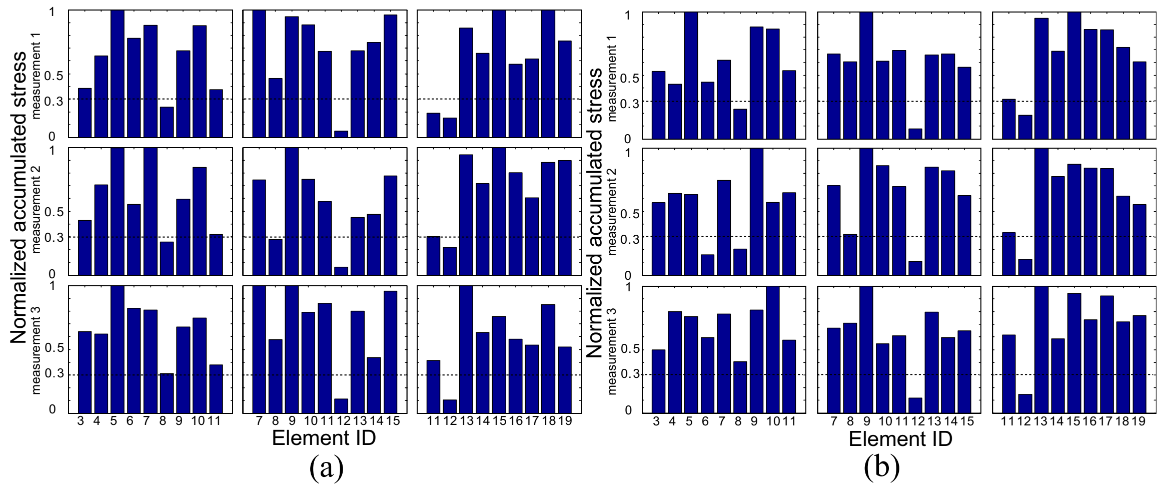
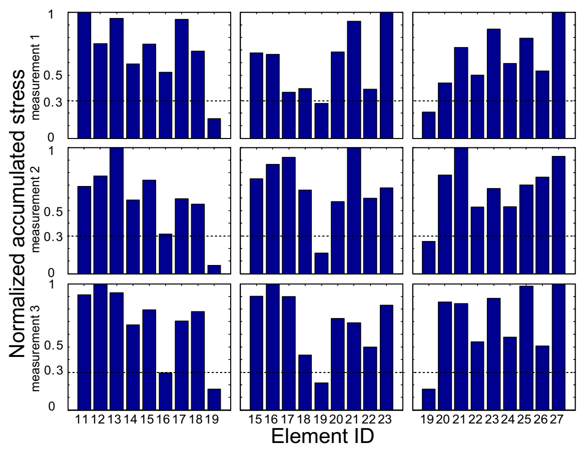


Figure 8.17. Damage detection (damaged element: Element 12): (a) mass perturbation DLV; and (b) SDLV.

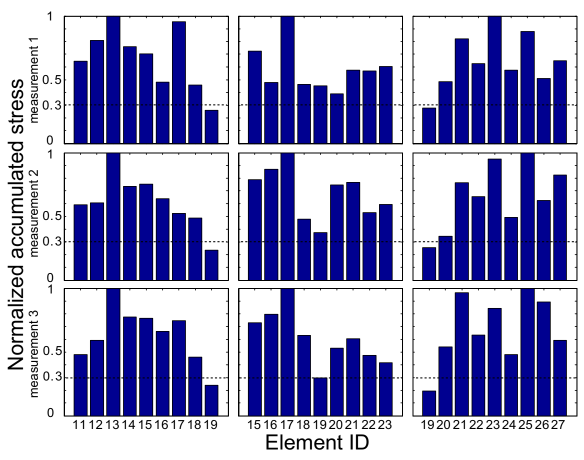
weight may explain the observed false-negatives. Nonlinearity of the structure and observation noise are also considered possible causes of the false-negatives.

One of the two DLV methods is not always better than the other in terms of damage detection capability. For example, damage detection on a vertical element, Element 11, in Figure 8.16 indicates the SDLV method performs better than the mass perturbation DLV method. On the other hand, damage on another vertical element, Element 19, in Figure 8.18 can be detected more reliably by the mass perturbation DLV method.

When the number of false-positives and false-negatives is counted, the SDLV method seems to experience fewer false damage detections. The number of false-positives and false-negatives is summarized in Tables 8.4. to 8.7. For example, elements 7, 8, 9, and 10

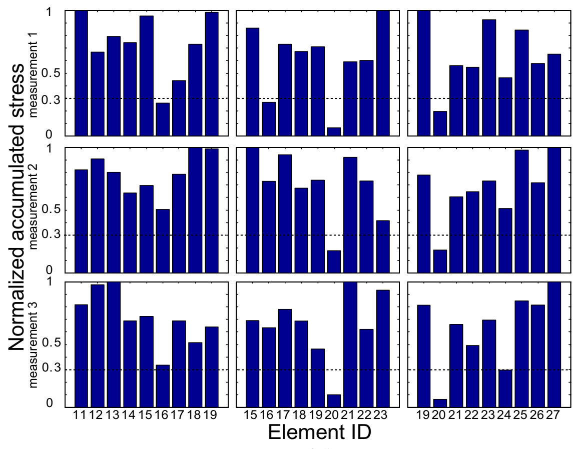


(a)

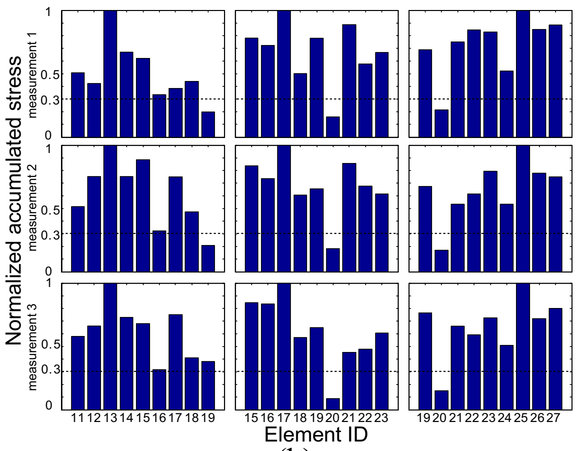


(b)

Figure 8.18. Damage detection (damaged element: Element 19): (a) mass perturbation DLV; and (b) SDLV.



(a)



(b)

Figure 8.19. Damage detection (damaged element: Element 20): (a) mass perturbation DLV; and (b) SDLV.

are commonly monitored by sensor communities 1 and 2, resulting in two damage detection cases per measurement. Repeating the measurement process three times results in six damage detection cases for each of these elements. When no false damage detection cases are observed out of six, the corresponding cells in these tables appear as “0/6.” The numerator represents the number of false damage detection cases, while the denominator represents the total number of damage detection cases. Elements 3, 4, 5, and 6 are monitored by only one community; thus, the denominator of the corresponding cells is three instead of six. Element 11 is monitored by three communities, so the denominator is nine. When element 8 is damaged, for example, the cell corresponding to this element represents false-negative damage detection while other cells in the same column represent

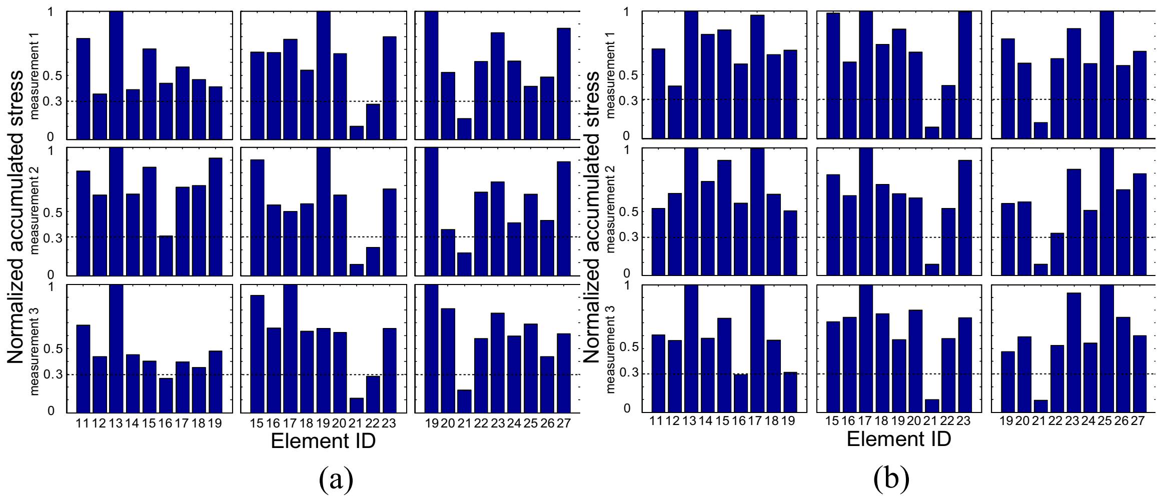


Figure 8.20. Damage detection (damaged element: Element 21): (a) mass perturbation DLV; and (b) SDLV.

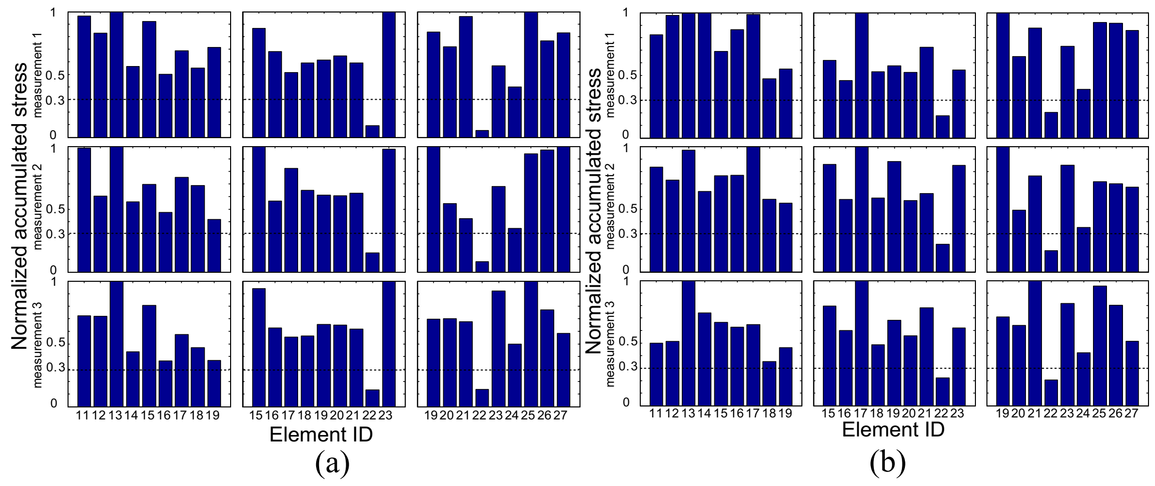


Figure 8.21. Damage detection (damaged element: Element 22): (a) mass perturbation DLV; and (b) SDLV.

false-positives. The cells corresponding to false-negatives are shaded to distinguish false-negatives from false-positives. The number of false damage detection using the mass perturbation DLV method is, overall, larger than the number for the SDLV method, though the sample size is small.

Tables 8.4 to 8.7 also suggest a way for robust damage detection. The summary of damage detection using the SDLV method in Tables 8.6 and 8.7 shows that the percentage of false damage detection cases is always smaller than or equal to 33 percent, while the percentage for the mass perturbation DLV method can be larger than 50 percent. From this observation, reliable damage localization might be possible by repeating damage detection with the SDLV method and by adopting the majority of damage detection results on each element. This conjecture needs to be confirmed with a larger number of samples.

Table 8.4. *Summary of False Damage Detection Using the Mass Perturbation DLV Method (Damaged Elements 8-12)*

Damaged Element	*8*		*9*		*10*		*11*		*12*	
	False Damage Detection /Total	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*
3	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
4	*0/3*	*0*	*1/3*	*33*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
5	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
6	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
7	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
8	*1/6*	*17*	*3/6*	*50*	*0/6*	*0*	*2/6*	*33*	*3/6*	*50*
9	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
10	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
11	*0/9*	*0*	*2/9*	*22*	*1/9*	*11*	*7/9*	*78*	*1/9*	*11*
12	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
13	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
14	*0/6*	*0*	*2/6*	*33*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
15	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
16	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
17	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
18	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
19	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*

8.9 Summary

In this chapter, Imote2s were installed on a truss model, and damage to the truss simulated by replacing an element with one of reduced cross section was identified by the proposed SHM framework. Required time and power consumption, as well as fault tolerance, were investigated. While some false-positive/negative damage detection cases were observed, damaged elements were identified in most cases. The proposed SHM framework employing DCS for SHM has, thus, been experimentally verified.

Table 8.5. *Summary of False Damage Detection Using the Mass Perturbation DLV Method (Damaged Elements 19-22)*

Damaged Element	*19*		*20*		*21*		*22*	
	False Damage Detection /Total	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*
11	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
12	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
13	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
14	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
15	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
16	*1/6*	*17*	*2/6*	*33*	*1/6*	*16*	*0/6*	*0*
17	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
18	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
19	*0/9*	*0*	*0/9*	*0*	*0/9*	*0*	*0/9*	*0*
20	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
21	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
22	*0/6*	*0*	*0/6*	*0*	*3/6*	*50*	*0/6*	*0*
23	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
24	*0/3*	*0*	*1/3*	*33*	*0/3*	*0*	*0/3*	*0*
25	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
26	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
27	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*

Table 8.6. *Summary of False Damage Detection Using the SDLV Method (Damaged Elements 8-12)*

Damaged Element	*8*		*9*		*10*		*11*		*12*	
	False Damage Detection /Total	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*
3	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
4	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
5	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
6	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*1/3*	*33*
7	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
8	*0/6*	*0*	*1/6*	*16*	*1/6*	*17*	*2/6*	*33*	*2/6*	*33*
9	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
10	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
11	*0/9*	*0*	*0/9*	*0*	*0/9*	*0*	*2/9*	*22*	*0/9*	*0*
12	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
13	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
14	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
15	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
16	*0/3*	*0*	*1/3*	*33*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
17	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
18	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
19	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*

Table 8.7. *Summary of False Damage Detection Using the SDLV Method (Damaged Elements 19-22)*

Damaged Element	*19*		*20*		*21*		*22*	
	False Damage Detection /Total	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*	*False Damage Detection /Total*	*%*
11	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
12	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
13	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
14	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
15	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
16	*0/6*	*0*	*0/6*	*0*	*1/6*	*16*	*0/6*	*0*
17	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
18	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
19	*2/9*	*22*	*2/9*	*22*	*0/9*	*0*	*0/9*	*0*
20	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
21	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
22	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
23	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*	*0/6*	*0*
24	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
25	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
26	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*
27	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*	*0/3*	*0*

Chapter 9

CONCLUSIONS AND FUTURE STUDIES

9.1 Conclusions

The research detailed in this report has established a framework for structural health monitoring on a network of smart sensors. This framework has been experimentally verified on networks of Imote2s, resulting in the realization of the first hierarchical smart sensor network for structural health monitoring. The structural health monitoring (SHM) system has essential features, such as scalability to a large number of smart sensors, promising damage detection capability, and autonomous operation. The software developed under this research effort is open-source and is available at: http://shm.cs.uiuc.edu/.

Background for this research was first provided. Smart sensors with computational and communication capabilities have been developed for various applications. These capabilities have been considered to offer new opportunities for the monitoring of structures. The inexpensive nature of smart sensors supports densely instrumenting structures; dense arrays of sensors have the potential to provide structural information at a level of detail never before available. However, smart sensor usage in SHM applications has encountered a number of difficulties. Many of these difficulties emanate from the lack of adequate resources on smart sensors. From a hardware perspective, smart sensors are usually battery-powered, and have limited RAM and relatively slow communication speed. Middleware services developed for such hardware are not necessarily suitable for SHM applications. Smart sensors have intrinsic synchronization error, and communication among sensors can be unreliable and/or slow. A well-developed smart sensor platform that can be directly used for SHM applications has only recently been reported. The Imote2 smart sensor platform will soon be released for resource demanding applications such as SHM of civil infrastructure. The substantially richer hardware resources on the Imote2, as compared with other smart sensors, better suits SHM applications. However, the Imote2 still misses some middleware services for realization of smart sensor SHM systems. Another type of difficulty regards algorithmic issues. SHM algorithms previously deployed on smart sensors have been based on either centralized data acquisition or independent data processing; obtaining both scalability and effective damage detection capabilities has been difficult. The Distributed Computing Strategy (DCS) for SHM was proposed by Gao (2005) as a promising SHM algorithm that can benefit from a large number of sensors. Data obtained at densely instrumented smart sensors are processed in local sensor communities in a coordinated and distributed manner. This SHM strategy, however, was previously demonstrated only on a PC with numerical simulation data or data from wired sensors. This research first studied and provided basic functionality necessary to implement the DCS on the Imote2s smart sensor platform.

be scalability, autonomous distributed computing, fault tolerance, etc. A system architecture that potentially provides these desirable characteristics was proposed as a homogeneous network of smart sensors consisting of Imote2s which run the DCS for SHM. Realization of this system was explained in the subsequent chapters.

The availability of appropriate sensors is essential for SHM applications. Sensor board customizability was demonstrated in Chapter 4. Strain is one of important physical quantities utilized in SHM applications. While accelerometers are often available on smart sensors, and their suitability for SHM applications has been studied, a smart sensor with strain sensors is rare. The flexibility of the smart sensor platforms was shown through the development of a strain sensor board equipped with a Wheatstone bridge circuit for the Berkeley Mote platform. The strain sensor board is an analog circuit, needing an antialiasing filter, which was also developed. Experimental verification of these sensor boards demonstrated the customizability of smart sensor boards.

Middleware services frequently needed in SHM applications were studied and realized on the Imote2. The amount of data required for SHM applications is usually so large that centrally collecting all of the data is impractical, if not impossible. A model-based data aggregation service was developed to estimate required correlation functions in a distributed manner. Data transfer requirements were greatly reduced. Another important middleware service developed was reliable communication. Lost communication packets carrying data can degrade signals in a similar way as observation noise does. The loss of a packet carrying a command may leave an SHM system in an unknown state. Reliable communication is, therefore, essential. Reliable communication protocols suitable for long data records and a short comments were each developed for both unicast and multicast applications. Synchronized sensing is also a crucial middleware service. Unsynchronized signals distort modal identification results, especially the phase of mode shapes. Even when clocks on smart sensors are synchronized, synchronized sensing is not necessarily guaranteed. When sensing commences cannot be easily controlled. To achieve synchronized sensing, clocks on the Imote2s were first synchronized, and then signals were acquired with time stamps; the signals were subsequently resampled based on the time stamps. In this way, signals were synchronized with each other with high precision. The developed middleware services allow implementation of DCS for SHM, as well as a wide variety of SHM applications, on Imote2s.

SHM algorithms implemented on Imote2s are based on DCS for SHM. In a local sensor community, NExT and ERA estimates the modal properties of a structure from acceleration responses. The mass perturbation DLV method localizes damaged element using modal properties identified before and after damage. The damage localization results are exchanged among neighboring local sensor communities to confirm the consistency of localization results in overlapping parts of sensor communities. If not, measurement and damage localization are repeated. This strategy was extended by replacing the mass perturbation DLV method with the SDLV method, which eliminates the need for mass normalization constant estimation.

Using the middleware services and algorithms, a framework for SHM using smart sensors was realized on the Imote2 platform. Numerical functions used in the DCS algorithms were first ported to Imote2s and their numerical accuracy and memory size requirements examined. Second, the sensing capability of the Imote2 was compared with conventional wired accelerometers. Third, using the ported numerical functions, the DCS algorithms were implemented. These algorithms were examined by injecting data from wired sensors into the memory of the Imote2s and processing the data both on the Imote2s and on a PC. Data processing on the Imote2s and the PC was shown to be numerically identical. An autonomous capability was realized in the system by assigning tasks to be executed to a one-byte variable and managing this variable. In this way, DCS for SHM was realized on Imote2s, and the validity of the implementation was examined component by component.

Finally, the smart sensor SHM system was experimentally verified. Ten Imote2s were installed on the three-dimensional truss structure, forming three local sensor communities with overlap. The outputs of each step of DCS for SHM were sent to the base station and compared with the corresponding outputs processed on a PC. The Imote2 networks performed modal analysis and damage localization as designed, and successfully localized simulated structural damage. Calculation and communication, as well as battery life of the SHM system, were then estimated. The results showed that the system does not need excessive time to perform data acquisition and processing and does not consume large amounts of power. However, the battery life needs to be improved if Imote2s are to be used for year-round monitoring. The damage detection capability was next examined for various damage cases. False damage detection was observed in some cases. These false-positives/negatives need to be studied and reduced/eliminated. Nonetheless, in most experiments, the SHM system was able to localize damage, demonstrating its damage detection capability.

This research provided the first realization of a hierarchical SHM system for smart sensors that is scalable to networks of densely distributed smart sensors. The system has good damage detection capability and can be operated autonomously. Battery life of the system is moderate. The next section describes suggested further study to extend the developed frame to be applicable to SHM of full-scale structures using a dense array of smart sensors.

9.2 Future studies

9.2.1 Sensing capability

As examined in section 7.2.1, acceleration signals from the Imote2s were shown to deviate from those measured by conventional accelerometers, even after the resampling process. For measured signals to be meaningful, the signals need to be reliable. The difference between the Imote2 signals and the reference sensor signals requires that the reliability of Imote2 sensors be inspected more in detail for future usage. Each of the components involved in sensing, such as accelerometer, time stamping, resampling, etc., needs to be examined carefully for the Imote2 to measure acceleration reliably.

For SHM of full-scale structures, the sensing capability in the low-frequency range is especially important, though this capability was not extensively studied in this research.

The lowest frequencies of interest for the truss structure used in the experiments were around 20 Hz, and the Imote2 sensing capability was shown to be sufficient for the DCS around these natural frequencies. However, natural frequencies of full-scale buildings and bridges are usually below 10 Hz. Three-dimensional sensing characteristics need to be examined in the low frequency range.

Future smart sensors used for ambient vibration measurement need to have better resolution. The least significant bit of the digital output of the Imote2 accelerometer is about 1 mg. Therefore, the resolution is at best 1 mg. While this resolution is sufficient for this research, ambient vibration measurement of civil infrastructure needs better resolution. An accelerometer chip with a low noise level and higher resolution is needed for ambient vibration measurement; if the accelerometer does not have digital outputs, the ADC also needs to have high resolution.

While smart sensors are often equipped with accelerometers, other types of sensors often utilized in civil engineering applications need to be developed for the Imote2. Though a strain sensor board for the Berkeley Mote has been developed, the Imote2 does not have a strain sensor board. A strain sensor board for the Imote2 will be beneficial to civil engineers.

Another problem is failure in sensing. Under the current design of the system, all of the sensors repeat sensing if one or more sensors fail to start sensing. The system can be modified so that sensing failure of a node makes only the sensors in the same community retake data. Even though the expected number of repetitions can be reduced in this way, this repetition should preferably be avoided by decreasing the chances of sensing failure.

9.2.2 Damage detection capability

The damage detection algorithms sometimes fail to detect damage. Repeated false damage detection was observed, though previous work on the DLV method did not report such false detection. Because the DCS for SHM on simulated numerical truss responses does not show false-negatives or repeated false-positive damage detection, such false damage detection in the Imote2 experiments is considered to be due primarily to causes specific to smart sensors. One possible cause is observation noise from imperfect measurements. As stated before, sensing hardware for the Imote2 still has room for improvement. Inaccurate sensing results in inaccurate damage detection. Another possible cause is a discrepancy between the physical truss and the model assumed in DCS for SHM. NExT, ERA, and the DLV methods all assume a linear model. If linearity is not satisfied, damage detection may become faulty. Errors in the numerical truss model used in the stress analysis possibly introduced inaccuracy in the damage detection, too. Before the application of the SHM system to the full-scale structures, the conditions under which damage detection becomes faulty need to be clarified, and the reliability needs to be improved.

The influence of other factors, such as temperature and humidity, should be accommodated in the damage detection strategy. For example, natural frequencies of a structure change along with temperature. The effects of temperature and humidity changes on the DLV methods have not yet been studied.

Integration of different types of information obtained from smart sensors has the possibility of improving damage detection capabilities. For example, smart sensors measuring acceleration, strain, wind velocity, temperature, and humidity may use four types of sensors to detect damage more reliably than smart sensors employing only accelerometers. Each measurand has its own characteristics. Acceleration responses reflect global motion of a structure and have particularly rich information at higher frequencies; on the other hand, strain is a local physical quantity, with its information concentrated at lower frequencies. Wind velocity affects the input force, damping, and stiffness. Temperature and humidity change at slower rate, affecting structural characteristics. Integration of different types of information may result in more reliable damage detection.

Furthermore, the applicability of the algorithms to nontruss structures needs to be examined and experimentally verified. The applicability has been studied mainly for truss structures. For DCS for SHM to be more widely used in monitoring civil infrastructure, the applicability to a wide range of structure types should be demonstrated.

9.2.3 Power harvesting

The three AAA batteries powering the Imote2 have a limited life. For some applications needing only a few samples taken infrequently, the batteries may last for months or even years. However, data-intensive applications such as SHM consume significant power, shortening the battery life. A battery life of years using only a few AAA batteries is not likely to be achieved in the near future. Using larger batteries is one practical solution when a small form factor is not required. The life of a smart sensor can be lengthened by increasing the battery capacity. Alternatively, power harvesting at the smart sensor nodes is a promising approach to achieve semipermanent monitoring of structures using nonplugged-in smart sensors.

Several energy sources can be identified for development of SHM power harvesting strategies. Wind energy is a potential energy source. Bridges are most likely constructed over a river, street, or railroad tracks where obstacles to block the wind are limited. Wind velocities at bridge cites are expected to be relatively high, making power harvesting with wind energy promising. Solar power is another candidate, though available energy depends on the climate at the site and sunlight availability of sensor locations. As opposed to wind power, solar power harvesting does not have any moving parts; therefore, vibration originating from solar power harvesting will not contaminate measurement of structural vibration signals. Another energy source, structural vibration energy, can conceptually be converted to electrical energy. Considering that vibration of civil infrastructure is normally in the low frequency range, power harvesting from such vibration might be difficult. Devices to capture small sources of energy and condition stored energy for smart sensor use have also been reported (Advanced Linear Devices, Inc., 2007). Further study is required to use these energy sources reliably for smart sensors.

Even when power harvesting is realized, power consumption on the smart sensor should be kept moderate. The amount of energy available from wind, solar power, or structural vibration is considered relatively small. Providing ample power to the smart sensors from these energy sources is not likely to happen soon. Power consumption at smart sensors, therefore, needs to be well-managed.

9.2.4 Power management and scheduling

Although the realized SHM system manages power consumption, the management still has room for improvement. The system does change the CPU speed depending on the tasks to be performed, and the system does prepare to enter a sleep mode when all of the tasks are performed. However, the change in CPU speed is only between two of four available speeds, and the Imote2 does not actually enter a sleep mode. By addressing these problems, power consumption can be further reduced.

In the future, DCS for SHM can be performed periodically. For example, the measurement and data processing can take place once every few days. While not active, the Imote2 can be in a sleep mode, only checking its clock. When the monitoring time approaches, the Imote2 can wake up and perform the necessary monitoring.

9.2.5 Monitoring of occasional events such as earthquakes

For monitoring of occasional events such as earthquakes, a scheme to wake up networks of smart sensors in a sufficiently short time is needed. Assigning one sensor node to monitor possible earthquakes continuously and to disseminate wake-up signals to all of the other nodes is one approach. However, the radio is turned off to save power in the sleep mode. Therefore, waking up all of the smart sensors by a wake-up command sent through the radio is not straightforward. Periodically turning on the radio to listen to a wake-up preamble is one possible solution, though such periodic usage of the radio consumes power. The interval of the periodic radio listening can be shortened to improve the response time in system wake-up at the expense of power consumption. If the start of an earthquake can be predicted, for example by measuring primary waves of an earthquake near the epicenter, the response time need not be so short, allowing relatively long intervals between the periodic radio listening. Another approach is to equip all of the sensor nodes with a mechanism to detect large motion and trigger sensing. A mechanism to detect large motion mechanically without power is preferable. Thus, system wake-up at the beginning of a rare event is a key issue.

9.2.6 Multihop

SHM systems to monitor full-scale structures may need multihop communication, while the SHM system realized in this research is based on single-hop communication. A sensor community for full-scale structures can be physically larger than the communication range for a smart sensor. Multihop communication is necessary in such cases. The unicast and multicast protocols need to be extended for multihop communication.

9.2.7 Communication range adjustment

When the communication range of a smart sensor includes neighboring sensor communities, the communication range should be shortened to reduce RF interference. When two sensor communities are out of communication range of each other, these communities can perform communication in parallel without interfering with each other. Communication range adjustment results in faster execution of DCS and smaller power consumption.

9.2.8 Reliability of the system

For the system to be used for years, the reliability of smart sensor nodes needs to be improved. In the current system, the Imote2s sometimes hangs up. The possibility of hang-ups should be reduced.

Additionally, node failure tolerance should preferably be realized, because the possibility of node failure cannot be completely eliminated; for example, even a PC with better resources sometimes hangs up. Failure of a single node should not provoke failure of the entire system.

Even when all of the nodes seem to be working, sensor signals may be erroneous. For example, sensors may not be firmly attached to a structure. Such Byzantine-type error can possibly be detected by checking the linearity between measured signals. When the linearity between two signals is much lower than that of other pairs, one of the two signals may come from a sensor which is not adequately installed. Implementation of such a Byzantine error detection algorithm can improve the reliability of the damage detection system.

9.2.9 Environmental hardening

The Imote2 needs to be packaged before the node can be installed on full-scale structures, especially when the Imote2 is used outside. The package needs to protect the Imote2 from weather, birds, insects, etc. Attention needs to be paid to not blocking radio reception/transmission. Also, the packaging should not absorb or amplify vibration.

9.2.10 Multiple purpose usage of smart sensors

Smart sensor networks can potentially serve multiple purposes, while the developed SHM framework currently serves only for SHM purposes. Smart sensors can be used for traffic monitoring, local weather monitoring, fire detection, etc. Integration of these services enhances the value of smart sensors and makes introduction of smart sensor systems more attractive from a cost-benefit perspective.

REFERENCES

Actis, R. L. & Dimarogonas, A. D. (1989). "Non-linear effects due to closing cracks in vibrating beams." *ASME Design Engineering Division Publication DE-Structural Vibration and Acoustics*, *18*(3), 99-104.

Adler, R., Flanigan, M., Huang, J., Kling, R., Kushalnagar, N., Nachman, L., Wan, C-Y., & Yarvis, M. (2005). "Intel mote 2: an advanced platform for demanding sensor network applications." *Proceedings of 3rd Int. Conference on Embedded Networked Sensor Systems,* 298-298.

Advanced Linear Devices, Inc. <http://www.aldinc.com> (Mar. 2, 2007).

Agardh, L. (1991). "Modal analysis of two concrete bridges." *Structural Engineering International*, *1*(4), 35-39.

Agre, J. R., Clare, L. P., Pottie, G. J., & Romanov, N. P. (1999). "Development platform for self-organizing wireless sensor networks." *Proceedings of SPIE - Conference on Unattended Ground Sensor Technologies and Applications*, Vol. 3713, 257-268.

Aktan, A. E., Lee, K. L., Chuntavan, C., & Aksel, T. (1994). "Modal testing for structural identification and condition assessment of constructed facilities." *Proceedings of International Modal Analysis Conference*, Honolulu, HI, (1), 464-468.

Alampalli, S., Fu, G., & Aziz, I. A. (1992). "Modal analysis as a bridge inspection tool." *Proceedings of 10th International Modal Analysis Conference*, San Diego, CA, 2, 1359-1366.

Al-Najjar, B. (2000). "Accuracy, effectiveness and improvement of vibration-based maintenance in paper mills; case studies." *Journal of Sound and Vibration*, *229*(2), 389-410.

Analog Devices, Inc. (2007). "AD623 - single supply, rail-rail, low cost instrumentation amplifiers." Datasheet, <http://www.analog.com/UploadedFiles/Data_Sheets/516895375AD623_c.pdf> (Mar. 2, 2007).

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., & Sorensen, D. (1999). *LAPACK User's Guide*, Philadelphia: Society for Industrial and Applied Mathematics.

Aoki, S., Fujino, Y., & Abe, M. (2003). "Intelligent bridge maintenance system using MEMS and network technology." *Proceedings of SPIE - Smart Systems and NDE for Civil Infrastructures*, 5057, 37-42.

Arici, Y. & Mosalam, K. M. (2003). "Modal analysis of a densely instrumented building using strong motion data." *Proceedings of Int. Conference on Applications of Statistics & Probability in Civil Engineering*, San Francisco, CA., 419-426.

Arms, S. P., Townsend, C. P., Churchhill, D. L., Hamel, M. J., Galbreath, J. H., & Mundell, S. W. (2004). "Frequency agile wireless sensor networks." *Proceedings of SPIE - Smart Electronics, MEMS, BioMEMS, and Nanotechnology*, 5389, 468-475.

Basheer, M. R., Rao, V., & Derriso, M. (2003). "Self organizing wireless sensor networks for structural health monitoring." *Proceedings of the 4th Int. Workshop on Structural Health Monitoring*, Stanford, CA, 1193-1206.

Becker, K. C., Byington, C. S., Forbes, N. A., & Nickerson G. W. (1998). "Predicting and preventing machine failures, *The Industrial Physicist*, American Institute of Physics, 20-23.

Begg, R. D., Mackenzie, A. C., Dodds, C. J., & Loland, O. (1976). "Structural integrity monitoring using digital processing of vibration signals." *Proceedings of 8th Annual Offshore Technology Conference*, Houston, TX, 305-311.

Bendat, J. S., & Piersol, A. G. (2000). *Random data: analysis and measurement procedures*, New York: Wiley.

Bernal, D. (2002). "Load vectors for damage localization." *Journal of Engineering Mechanics*, *128*(1), 7-14.

Bernal, D. (2004). "Modal scaling from known added masses." *Journal of Engineering Mechanics*, *130*(9), 1083-1088.

Bernal, D. (2006). "Flexibility-based damage localization from stochastic realization results." *Journal of Engineering Mechanics*, *132*(6), 651-658.

Bernal, D. & Gunes, B. (2004). "Flexibility based approach for damage characterization: benchmark application." *Journal of Engineering. Mechanics*, *130*(1), 61-70.

Bhatti, S. Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., & Han, R. (2005). "MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms." *Mobile Networks and Applications*, *10*(4), 563-579.

Bishop, C. M. (1994). "Neural networks and their applications." *Review of Scientific Instruments*, *65*, 1803-1832.

Biswas, M., Pandey, A. K. & Samman, M. M. (1990). "Diagnostic Experimental Spectral/ Modal Analysis of a Highway Bridge." *Int. Journal of Analytical and Experimental Modal Analysis*, *5*, 33-42.

Brincker, R., Zhang, L., & Andersen, P. (2001). "Modal identification of output-only systems using frequency domain decomposition." *Smart Materials and Structures*, *10*, 441-445.

Bult, K., Burstein, A., Chang, D., Dong, M., Fielding, M., Kruglick, E., Ho, J., Lin, F. Lin, T. H., Kaiser, W. J., Mukai, R., Nelson, P., Newburg, F. L., Pister, K. S. J., Pottie, G., Sanchez, H., Stafsudd, O. M., Tan, K. B., Ward, C. M., Yung, G., Xue, S., Marcy, H. & Yao, J. (1996). "Wireless integrated microsensors." *Technical Digest of the 1996 Solid State Sensor and Actuator Workshop*, Transducers Research Foundation, Hilton Head Island, SC, 205-210.

Casciati, F., Faravelli, L, Borghetti, F., & Fornasari, A. (2003). "Future trends in infrastructure monitoring." *Structural Health Monitoring and Intelligent Infrastructure*, *2*, 997-1002.

Casciati, F., Casciati, S., Faravelli, L., & Rossi, R. (2004). "Hybrid wireless sensor network." *Proceedings of SPIE - Smart Structures and Materials: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, Berkeley, CA, 5391, 308-313.

Celebi, M. (2002). "Seismic instrumentation of buildings (with emphasis on federal buildings)" *Special GSA/USGS Project, an administrative report*, United States Geological Survey, Menlo Park, CA.

Chance, J., Tomlinson, G. R., & Worden, K. (1994). "A simplified approach to the numerical and experimental modeling of the dynamics of a cracked beam." *Proceedings of 12th International Modal Analysis Conference*, 778-785.

Chen, M., Glaser, S. D., & Oberheim, T. (2006). "Terra-Scope - A MEMS-based Vertical Seismic Array." *Smart Structures & Systems*, *2*(2), 115-126.

Chintalapudi, K., Paek, J., Gnawali, O., Fu, T., Dantu, K., Caffrey, J., Govindan, R., & Johnson, E. (2006). "Structural damage detection and localization using NetSHM." *Proceedings of 5th Int. Conference on Information Processing in Sensor Networks.* Nashville, TN.

Chipcon, Inc. <http://www.chipcon.com> (Mar. 2, 2007).

Chowdhury, M. R. (1990). "Experimental modal testing and analysis of fracture damage in structures by the modal frequency method." *Proceedings of 8th International Modal Analysis Conference*. Florida, (1), 109-114.

Chung, H.-C., Enomoto, T., Shinozuka, M., Chou, P., Park, C., Yokoi, I., & Morishita, S. (2004). "Real time visualization of structural response with wireless MEMS sensors." *Proceedings of 13th World Conference on Earthquake Engineering*, No. 121, 1-10.

Clayton, E. H., & Spencer, B. F., Jr. (2001). "Development of an experimental model for the study of infrastructure preservation." *Proceedings of the National Conf. on Undergraduate Research (NCUR)*, Whitewater, WI.

Collacott, R. A. (1977) *Mechanical fault diagnosis and condition monitoring*, London: Chapman and Hall.

Cooley, J. M. & Tukey, J. W. (1965). "An algorithm for the machine calculation of complex Fourier series." *Mathematics of Computation*, *19*:297-301.

Crossbow Technology, Inc. <http://www.xbow.com> (Mar. 2, 2007).

Danielson, G. C. & Lanczos, C. (1942). "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids." *Journal of the Franklin Institute, 233*, 365-380 and 435-452.

DataGrid Project. <http://eu-datagrid.web.cern.ch/eu-datagrid/default.htm> (Mar. 2, 2007).

Doebling, S. W. (1995). Measurement of structural flexibility matrices for experiments with incomplete reciprocity, (Doctoral dissertation, University of Colorado, 1995).

Doebling, S. W. & Farrar, C. R. (1999). *The state of the art in structural identification of constructed facilities*, A report by American Society of Civil Engineers Committee on Structural Identification of Constructed Facilities.

Doebling, S. W., Farrar, C. R., & Prime, M. B. (1998). "A summary review of vibration-based damage identification methods." Shock Vib. Dig., 30(2), 91-105.

Doebling, S. W., Farrar, C. R., Prime, M. B., & Shevitz, D. W. (1996). "Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review." Los *Alamos National Laboratory Report*, LA-13070-MS.

Doherty, L., Pister, K. S. J., & Ghaoui. L. E. (2001). "Convex position estimation in wireless sensor networks." *Proceedings of the IEEE Infocom*, AK., 1655-1663.

Dubin, E. E. & Yanev, B. S. (2001). "Managing the East river bridges in New York City." *Proceedings of SPIE - 7th International Symposium on Smart Structures and Materials*, Newport, CA, 60-74.

Dust Networks. <http://www.dust-inc.com> (Mar. 2, 2007).

Electronics Services Shop. <http://www.ece.uiuc.edu/eshop> (Mar. 2, 2007).

Elson, J., Girod, L., & Estrin, D. (2002). "Fine-grained network time synchronization using reference broadcasts." *Proceedings of 5th Symposium on Operating Systems Design and Implementation*, Boston, MA.

Ember Corporation. <http://www.ember.com> (Mar. 2, 2007).

EYES. <http://www.eyes.eu.org> (Mar. 2, 2007).

Farrar, C. R. (2001). "Historical overview of structural health monitoring." *Lecture Notes on Structural Health Monitoring using Statistical Pattern Recognition*. Los Alamos Dynamics, NM.

Farrar, C. R., Allen, D. W., Ball, S., Masquelier, M. P., & Park, G. (2005). "Coupling sensing hardware with data interrogation software for structural health monitoring." *Proceedings of 6th International Symposium on Dynamic Problems of Mechanics*, Ouro Preto, Brazil.

Farrar, C. R., Baker, W. E., Bell, T. M., Cone, E. M., Darling, T. W., Duffey, T. A., Eklund, A., & Migliori, A. (1994). "Dynamic characterization and damage detection in the I-40 bridge over the Rio Grande." *Los Alamos National Laboratory Report LA-12767-MS.*

Farrar, C. R., Sohn, H., Hemez, F. M., Anderson, M. C., Bement, M. T., Cornwell, P. J., Doebling, S. W., Lieven, N., Robertson, A. N., & Schultze, J. F. (2003). "Damage prognosis: current status and future needs." *Los Alamos National Laboratory Report LA-14051-MS.*

Feldman, M. & Braun, S. (1995). “Identification of nonlinear system parameters via the instantaneous frequency: application of the Hilbert transform and Wigner-Wille techniques.” *Proceedings of 13th International Modal Analysis Conference*, 637-642.

Fox, C. H. J. (1992). “The location of defects in structures: a comparison of the use of natural frequency and mode shape data.” *Proceedings of 10th International Modal Analysis Conference*, 522-528.

Galbreath, J. H., Townsend, C. P., Mundell, S. W., Hamel, M. J., Esser, B., Huston, D., & Arms, S. W. (2003). “Civil structure strain monitoring with power-efficient, high-speed wireless sensor networks.” *Proceedings of 4th Int. Workshop on Structural Health Monitoring*, 1215-1222.

Ganeriwal, S., Kumar, R., & Srivastava, M. B. (2003). “Timing-sync protocol for sensor networks.” *Proceedings of 1st International Conference On Embedded Networked Sensor Systems*, Los Angeles, CA., 138-149.

Gao, Y. (2005). Structural health monitoring strategies for smart sensor networks (Doctoral dissertation, University of Illinois at Urbana-Champaign, 2005).

Gao, Y., Spencer, B. F., Jr. & Ruiz-Sandoval, M. (2005). "Distributed computing algorithm for structural health monitoring." Structural Control and Health Monitoring, in review.

Gardner-Morse, M. G. & Huston, D. R. (1993). “Modal identification of cable stayed pedestrian bridge.” *Journal of Structural Engineering*, ASCE, *119*(11), 3384-3404.

Gay, D., Levis, P. Behren, R., Welsh, M., Brewer, E. & Culler, D. (2003). “The nesC language: a holistic approach to networked embedded systems.” *Proceedings of Programming Language Design and Implementation*.

Girod, L., Elson, J., Cerpa, A., Stathopoulos, T., Ramanathan, N., & Estrin, D. (2004). “EmStar: a software environment for developing and deploying wireless sensor networks.” *Proceedings of USENIX ‘04*, Boston, MA, 283-296.

Golden Gate Bridge, Highway and Transportation District. <http://goldengatebridge.org> (Mar. 2, 2007).

Gros, X. E. (1997). *NDT data fusion*, London: Arnold.

He, J. & Ewins, D. J. (1986). “Analytical stiffness matrix correction using measured vibration modes.” *Int. Journal of Analytical and Experimental Modal Analysis*, *1*(3), 9-14.

Hearn, G. & Testa, R. B. (1991). “Modal analysis for damage detection in structures.” *Journal of Structural Engineering*, *117*(10), 3042-3063.

Hermans, L. & Auweraer, H. V. (1999). “Modal testing and analysis of structures under operational conditions: industrial applications.” *Mechanical Systems and Signal Processing*, *13*(2), 193-216.

Hill, J., & Culler, D. (2002). “Mica: A wireless platform for deeply embedded networks.” *IEEE Micro.*, *22*(6), 12-24.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., & Pister, K. (2000). "System architecture directions for networked sensors." *Proceedings of 9th International Conference Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, 93-104.

Hollar, S. (2000). "COTS Dust." Master's Thesis, University of California, Berkeley, CA.

Hou, T.-C., Lynch, J. P., & Parra-Montesinos, G. (2005). "In-situ wireless monitoring of fiber reinforced cementitious composite bridge piers." *Proceedings of Int. Modal Analysis Conference*, Orlando, FL.

Ibrahim, S. R. & Mikulcik, E. C. (1977). "A method for the direct determination of vibration parameters from the free responses." *Shock and Vibration Bulletin*, *47*(4), 183-198.

Intel Corporation. <http://www.intel.com> (Mar. 2, 2007).

James, G. H., Carne, T. G., & Lauffer, J. P. (1993). "The natural excitation technique for modal parameter extraction from operating wind turbine,"*Report No. SAND92-1666*, *UC-261*, Sandia National Laboratories, NM.

James, G. H., Carne, T. G., Lauffer, J. P. & Nord, A. R. (1992). "Modal testing using natural excitation," *Proceedings of 10th Int. Modal Analysis Conference*, San Diego, CA.

Juang, J.-N. and Pappa, R. S. (1985). "An eigensystem realization algorithm for modal parameter identification and model reduction." *Journal of Guidance, Control, and Dynamics*, *8*(5):620-627.

Kaouk, M. & Zimmerman. D. C. (1994). "Structural damage assessment using a generalized minimum rank perturbation theory." *AIAA Journal*, *32*(4), 836-842.

Kaouk, M. & Zimmerman, D. C. (1995). "Reducing the required number of modes for structural damage assessment." *Proceedings of 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2802-2812, AIAA-95-1094-CP.

Karbhari, V. M., Guan, H. & Sikorsky, C. (2003). "Web-based structural health monitoring of a FRP composite bridge." *Structural Health Monitoring and Intelligent Infrastructure*, *1*, 217-225.

Kawahara, Y., Minami, M., Morikawa, H., & Aoyama, T. (2003). "Design and implementation of a sensor network node for ubiquitous computing environment." *Proceedings of IEEE Semiannual Vehicular Technology Conference*, Orlando, FL., 5, 3005-3009.

Kiremidjian, A. S., Straser, E. G., Meng, T. H., Law, K. & Soon, H. (1997). "Structural damage monitoring for civil structures." *Proceedings of Int. Workshop on Structural Health Monitoring*, Stanford, CA, 371-382.

Kirianaki, N. V., Yurish, S. Y., & Shpak, N. O. (2000). "Smart sensors with frequency output: state-of-the-art and future development." *Proceedings of IFAC Workshop on Progrmmable Devices and Systems*, Ostrava, Czech Republic, 37-42.

Kling, R. (2003). “Intel Mote: an enhanced sensor network node.” *Proceedings of Int. Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures*, Tokyo, Japan.

Kling, R., Adler, R., Huang, J., Hummel, V., & Nachman, L. (2005). “Intel Mote-based sensor networks.” *Structural Control and Health Monitoring, 12*, 469-479.

Ko, J. M., & Ni, Y. Q. (2005). “Technology developments in structural health monitoring of large-scale bridges.” *Engineering Structures*, *27*(12), 1715-1725.

Kottapalli, V. A., Kiremidjian, A. S., Lynch, J. P., Carryer, E., Kenny, T. W., Law, K. H., & Lei, Y. (2003). “Two-tiered wireless sensor network architecture for structural health monitoring.” *Proceedings of SPIE - The International Society for Optical Engineering - Smart Structures and Materials*, 5057, 8-19.

Krawczuk, M. & Ostachowicz, W. M. (1992). “Parametric vibrations of a beam with crack.” *Archive of Applied Mechanics, 62*, 463-473.

Kurata, N., Spencer, B. F., Jr., & Ruiz-Sandoval, M. (2004). “Building risk monitoring using wireless sensor network.” *Proceedings of 13th World Conference on Earthquake Engineering*, Vancouver, BC., Canada, No. 1406.

Kwon, Y., Mechitov, K. A., Sundresh, S., Kim, W., & Agha, G. A. (2005a). “Resilient localization for sensor networks in outdoor environments.” *Proceedings of 25th IEEE International Conference on Distributed Computing Systems*, Colombus, OH, 643-652.

Kwon, Y., Sundresh, S., Mechitov, K. A., & Agha, G. A. (2005b). "ActorNet: an actor platform for wireless sensor networks." *University of Illinois at Urbana-Champaign Technical Report.*

Law, Y., Dulman, S., Etalle, S., & Havinga, P. (2002). “Assessing security-critical energy-efficient sensor networks.” *Technical Report TR-CTIT-02-18*, Centre for Telematics and Information Technology, University of Twente, Netherlands.

Lieven, N. A. J. & Waters, T. P. (1994). “Error location using normalised orthogonality.” *Proceedings of 12th International Modal Analysis Conference*, Honolulu, HI, 1, 761-764.

Lin, C. S. (1990). “Location of modeling errors using modal test data.” *AIAA Journal*, *28*, 1650-1654.

Lin, R. M. & Ewins, D. J. (1990). “On the location of structural nonlinearity from modal testing - a feasibility study.” *Proceedings of 8th International Modal Analysis Conference*, 358-364.

Linear Technology. (2007). “LTC1682 - doubler charge pumps with low noise linear regulator.” Datasheet, <http://www.linear.com/pc/downloadDocument.do?navId=H0,C1,C1003,C1039,C1133,P1749,D1952> (Mar. 2, 2007).

Lynch, J. P. (2006). “Design of a wireless active sensing unit for localized structural health monitoring.” *Structural Control and Health Monitoring*, in press.

Lynch, J. P., Law, K. H., Kiremidjian, A. S., Carryer, E., Kenny, T. W., Patridge, A., & Sundararajan, A. (2002). “Validation of a wireless modular monitoring system for structures.” *Proceedings of SPIE - Smart Structures and Materials: Smart Systems for Bridges, Structures, and Highways*, San Diego, CA, 4696(2), 17-21.

Lynch, J. P. & Loh, K. (2006). “A summary review of wireless sensors and sensor networks for structural health monitoring.” *Shock and Vibration Digest*, in press.

Lynch, J. P., Sundararajan, A., Law, K. H., Kiremidjian, A. S., Carryer, E., Sohn, H., & Farrar, C. R. (2003). “Field validation of a wireless structural health monitoring system on the alamosa canyon bridge.” *Proceedings of SPIE - Smart Structures and Materials: Smart Systems and Nondestructive Evaluation for Civil Infrastructures*, 5057, 267-278.

Lynch, J. P., Wang, Y., Law, K. H., Yi, J.-H., Lee, C. G., & Yun, C. B. (2005). “Validation of a large-scale wireless structural monitoring system on the Geumdang bridge.” *Proceedings of the Int. Conference on Safety and Structural Reliability*, Rome, Italy.

Maia, N. M. M., & Silva, J. M. M. (1997). *Theoretical and experimental modal analysis.* Taunton Somerset, England: Research Studies Press.

Maroti, M. Kusy, B., Simon, G., & Ledeczi, A. (2004). “The flooding time synchronization protocol.” *Proceedings of 2nd International Conference On Embedded Networked Sensor Systems*, Baltimore, MD, 39-49.

Mason, A., Yazdi, N., Najafi, K., & Wise, K. (1995). “A low-power wireless microinstrumentation system for environmental monitoring.” *Dig. 8th Int. Conference on Solid-State Sensors and Actuators*, Stockholm, Sweden, 107-110.

Maxim Integrated Products, Inc. (2007). “Single/dual/quad, wide-bandwidth, low-power, single-supply, rail-to-rail i/o op amps.” *Datasheet*, <http://pdfserv.maxim-ic.com/en/ds/MAX4130-MAX4134.pdf> (Mar. 2, 2007).

Mechitov, K. A., Kim, W., Agha, G. A., & Nagayama, T. (2004). “High-frequency distributed sensing for structure monitoring.” *Proceedings of 1st Int. Workshop on Networked Sensing Systems*, Tokyo, Japan, 101–105.

MicroStrain, Inc. <http://www.microstrain.com> (Mar. 2, 2007).

Millennial Net. <http://www.millennial.net> (Mar. 2, 2007).

Miller, B. (2001). “Bluetooth and other wireless technologies.” Prentice Hall. <http://www.phptr.com/articles/printerfriendly.asp?p=24265> (Mar. 2, 2007).

Moore, M., Phares, B., Graybeal, B., Rolander, D., & Washer, G. (2001). “Reliability of visual inspection of highway bridges.” *FHWA Report No. FHWA-RD-01-020*, FHWA, U.S. Dept. of Transportation, Washington, DC.

Morgan, B. J. & Osterle R. G. (1985). “On-site modal analysis - a new powerful inspection technique.” *Proceedings of 2nd International Bridge Conference*, Pittsburg, PA, 108-114.

Mufti, A. A. (2003). “Integration of sensing in civil structures: development of the new discipline of civionics.” *Structural Health Monitoring and Intelligent Infrastructure*, *1*, 119-129.

Nagayama, T., Abe, M., Fujino, Y., & Ikeda, K. (2005). “Structural identfication of a non-proportionally damped system and its application to a full-scale suspension bridge.” *Journal of Structural Engineering*, *131*(10), 1536-1545.

Nagayama, T., Rice, J. A., & Spencer, B. F., Jr. (2006a). “Efficacy of Intel's Imote2 wireless sensor platform for structural health monitoring applications.” *Proceedings of Asia-Pacific Workshop on Structural Health Monitoring*, Yokohama, Japan.

Nagayama, T., Ruiz-Sandoval, M., Spencer, B. F., Mechitov K. A., & Agha, G. A. (2004). “Wireless strain sensor development for civil infrastructure.” *Proceedings of 1st Int. Workshop on Networked Sensing Systems*, Tokyo, Japan, 97-100.

Nagayama, T. Sim, S-H., Miyamori, Y., & Spencer, B. F., Jr. (2007). “Issues in structural health monitoring employing smart sensors.” *Smart Structures and Systems* (accepted).

Nagayama, T., Spencer, B. F., Jr., Agha, G. A., & Mechitov, K. A. (2006b), “Model-based data aggregation for structural monitoring employing smart sensors”, *Proceedings of 3rd Int. Conference on Networked Sensing Systems (INSS 2006)*, Chicago, IL, 203-210.

Nakamura, Y. (2004). “UrEDAS, urgent earthquake detection and alarm system, now and future.” *Proceedings of 13th World Conference on Earthquake Engineering*, Vancouver, B.C., Canada, Paper No. 908.

Nakamura, M., Masri, S. F., Chassiakos, A. G., & Caughey, T. K. (1998). “A method for non-parametric damage detection through the use of neural networks.” *Earthquake Engineering and Structural Dynamics*, *27*, 997-1010.

Nashif, A. D., Jones, D. I., & Henderson, J. P. (1985). *Vibration damping*. New York: Willey.

National Semiconductor. (1991). “A basic introduction to filtersdactive, passive, and switched-capacitor.” *National Semiconductor Application Note 779*, <http://www.national.com/an/AN/AN-779.pdf> (Mar. 2, 2007).

Nitta, Y., Nagayam, T., Spencer, B. F. Jr., & Nishitani, A. (2005). “Rapid damage assessment for the structures utilizing smart sensor MICA2 MOTE.” *Proceedings of 5th Int. Workshop on Structural Health Monitoring*, Stanford, CA, 283-290.

Nwosu, D. I., Swamidas, A. S. J., Guigne, J. Y., & Olowokere, D. O. (1995). “Studies on influence of cracks on the dynamic response of tubular T-joints for nondestructive evaluation.” *Proceedings of 13th International Modal Analysis Conference*, 1122-1128.

Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1999). *Discrete-time signal processing*. Upper Saddle River, NJ: Prentice Hall.

Oshima, T., Rahman, M. S., Mikami, S., Yamazaki, T., Takada, N., Lesko, J. J., & Kriz, R. D. (2000). "Application of smart materials and systems to long-term bridge health monitoring." *Proceedings of SPIE - Nondestructive Evaluation of Highways, Utilities, and Pipelines*, 3995, 253-263.

Ou, J. P., Li, H. W., Xiao, Y. Q., & Li, Q. S. (2005). "Health dynamic measurement of tall building using wireless sensor network." *Proceedings of SPIE - Smart Structures and Materials*, *5765*(1), 205-215.

Pandey, A. K., Biswas, M. & Samman, M. M. (1991). "Damage detection from changes in curvature mode shapes." *Journal of Sound and Vibration*, *145*(2), 321-332.

PCBExpress. <http://www.pcbexpress.com> (Mar. 2, 2007).

Pines, D. & Aktan, A. E. (2002). "Status of structural health monitoring of long-span bridges in the United States." *Progress in Structural Engineering and Materials*, *4*(4), 372-380.

Polastre, J., Szewczyk, R. & Culler, D. (2005). "Telos: enabling ultra-low power wireless research." *Proceedings of 4th International Conference on Information Processing in Sensor Networks*, Los Angeles, CA.

Press W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: the art of scientific computing*. New York: Cambridge University Press.

Rahimi, M., Hardik, S., Sukhatme, G. S., Heideman, J., & Deborah, E. (2003). "Studying the feasibility of energy harvesting in a mobile sensor networks." *Proceedings of IEEE Int. Conference on Robotics and Automation*, Taipai, Taiwan, 1, 19-24.

Romer, K., Kasten, O., & Mattern, F. (2002). "Middleware Challenges for Wireless Sensor Networks." *Mobile Computing and Communications Review*, 6(4), 59-61.

Ruiz-Sandoval, M. (2004). "Smart sensors" for civil infrastructure systems (Doctoral dissertation, University of Notre Dame, 2004).

Ruiz-Sandoval, M., Nagayama, T., & Spencer, B. F., Jr. (2006). "Sensor development using Berkeley Mote platform." *Journal of Earthquake Engineering*, *10*(2), 289-309 .

Salawu, O. S. (1997). "Detection of structural damage through changes in frequency: a review." *Engineering Structures*, *19*(9), 718-723.

Salawu, O. S. & Williams, C. (1994). "Damage location using vibration modeshapes." *Proceedings of 12th International Modal Analysis Conference*, 933-939.

Salawu, O. S. & Williams, C. (1995). "Review of full-scale dynamic testing of bridge structures." *Engineering Structures 17*(2), 113-121.

Sallen, R. P. & Key, E. L. (1955). "A practical method of designing active filters," *IRE Transactions on Circuit Theory*, CT-2, 74-85.

Satpathi, D., Victor, J. P., Wang, M. L., & Yang, H. Y. (1999). "Development of a PVDF film sensor for infrastructure monitoring." *Proceedings of SPIE - 6th Annual Int. Symposium on Smart Structures and Materials*, Newport Beach, CA, 3671, 90-99.

Sazonov, E., Janoyan, K., & Jha, R. (2004). "Wireless intelligent sensor network for autonomous structural health monitoring." *Proceedings of SPIE - Smart Structures and Materials: Smart Sensor Technology and Measurement Systems*, 5384, 305-314.

Screaming Circuit. <http://www.screamingcircuits.com> (Mar. 2, 2007).

Sensametrics. <http://www.sensametrics.com> (Mar. 2, 2007).

Sensicast Systems. <http://www.sensicast.com> (Mar. 2, 2007).

Shinozuka, M. (2003). "Homeland security and safety." *Proceedings of Structural Health Monitoring and Intelligent Infrastructure*, Tokyo, Japan, 1139-1145.

Silicon Designs, Inc. (2007). "Model 1221 - Low noise analog accelerometer." Datasheet, <http://silicondesigns.com/Pdf/1221.pdf> (Mar. 2, 2007).

Simon G., Maroti, M., & Ledeczi, A. (2004). "Sensor network-based countersniper system." *Proceedings of SenSys '04*, Baltimore, MD.

Skolnick, D. & Levine, N. (1997). "Why Use DSP? Digital signal processing 101- an introductory course in DSP system design:" *Analog Dialogue*, 31(1), <http://www.analog.com/library/analogDialogue/archives/31-1/DSP.html> (Mar. 2, 2007).

Sohn, H., Farrar, C. R., Hemez, F. M., Shunk, D. D., Stinemates, D. W., & Nadler B. R. (2003). "A review of structural health monitoring literature: 1996-2001" *Los Alamos National Laboratory Report, LA-13976-MS*.

Sohn, H., Park, H. W., Law, K., & Farrar, C. R. (2007). "Combination of a time reversal process and a consecutive outlier analysis for baseline-free damage diagnosis." *Journal of Intelligent Material Systems and Structures, 18*(4), 335-346.

Sohn, H., Worden, K., & Farrar, C. R. (2002). "Statistical damage classification under changing environmental and operational conditions." *Journal of Intelligent Material Systems and Structures*, *13*, 561-574.

Spectral Dynamics, Inc. <http://www.spectraldynamics.com> (Mar. 2, 2007).

Spencer, B. F., Jr. & Nagayama, T. (2006). "Smart sensor technology: A new paradigm for structural health monitoring." *Proceedings of Asia-Pacific Workshop on Structural health Monitoring*, Yokohama, Japan.

Srinivasan, M. G. & Kot, C. A. (1992). "Effects of damage on the modal parameters of a cylindrical shell." *Proceedings of 10th International Modal Analysis Conference*, 529-535.

Staszewski, W. J. (1998). "Structural and mechanical damage detection using wavelets." *The Shock and Vibration Digest, 30*(6), 457-472.

Stewart, R. W. & Pfann, E. (1998). "Oversampling and sigma-delta strategies for data conversion." *Electronics & Communication Engineering Journal*, *10*(1), 37-47.

STMicroelectronics. <http://www.st.com/stonline> (Mar. 2, 2007).

Straser, E. G. & Kiremidjian A. S. (1996). "A modular visual approach to damage monitoring for civil structures." *Proceedings of SPIE - Smart Structures and Materials*, San Diego, CA, *2719*, 112-122.

Straser, E. G. & Kiremidjian A. S. (1998). "A modular, wireless damage monitoring system for structures." *The John A. Blume Earthquake Engineering Center Technical Report*, *128*.

Studer, M. & Peters, K. (2004). "Multi-scale sensing for damage identification", *Smart Materials and Structures*, *13*, 283-294.

Sun, C. T. and Lu, Y. P. (1995). *Vibration damping of structural elements*. Englewood Cliffs, NJ: Prentice Hall.

Tang, J. P. and Leu, K. M. (1989). "Vibration tests and damage detection of P/C bridges" ICOSSAR 1989, *Proceedings of 5th International Conference on Structural Safety and Reliability*, ASCE, San Francisco, CA, 2263-2266.

Tanner, N. A., Farrar, C. R., & Sohn, H. (2002). "Structural health monitoring using wireless sensing systems with embedded processing." *Proceedings of SPIE - NDE and Health Monitoring of Aerospace Materials and Civil Infrastructure*, 4704, 215-224.

Tanner, N. A., Wait, J. R., Farrar, C. R., & Sohn, H. (2003). "Structural health monitoring using modular wireless sensors." *Journal of Intelligent Material Systems and Structures*, *14*(1), 43-56.

Texas Instruments. (2007) "Is DSP Right for You?" <http://focus.ti.com/dsp/docs/dspsupporto.tsp?sectionId=4&tabId=1443#what_why> (Mar. 2, 2007).

The MathWorks, Inc. <http://www.mathworks.com> (Mar. 2, 2007).

TinyOS. <http://www.tinyos.net> (Mar. 2, 2007).

Toksoy, T. & Aktan, A. E. (1994). "Bridge-condition assessment by modal flexibility." *Experimental Mechanics*, *34*, 271-278.

Vapnik, V. (1998). *Statistical Learning Theory*, New York: Wiley.

Wang, M. L., Gu, H., Lloyd, G. M., & Zhao, Y. (2003). "A multi-channel wireless PVDF displacement sensor for structural monitoring." *Proceedings of Int. Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures*, Tokyo, Japan, 1-6.

West, W. M. (1984). "Illustration of the use of modal assurance criterion to detectstructural changes in an orbiter test specimen." *Proceedings of Air Force Conference on Aircraft Structural Integrity*, 1-6.

Williams, C. & Salawu, O. S. (1997). "Damping as a damage indication parameter." *Proceedings of 15th International Modal Analysis Conference*, 1531-1536.

Wong, K.-Y. (2004). "Instrumentation and health monitoring of cable-supported bridges." *Structural Control and Health Monitoring*, 11(2), 91-124.

Yi, J.-H. & Yun, C.-B. (2004). “Comparative study on modal identification methods using output-only information.” *Structural Engineering and Mechanics*, *17*(3-4), 445-466.

Yurish, S. Y. (2005) “Intelligent opto sensors’ interfacing based on universal frequency-to-digital converter.” *Sensors and Transducers Magazine*, *56*(6), 326-334.

Zhang, Z. & Atkan, A. E. (1995). “The damage indices for constructed facilities.” *Proceedings of 13th International Modal Analysis Conference*, 1520-1529.

Zhao, F. & Guibas, L. J. (2004). *Wireless sensor networks: an information processing approach*. San Francisco, CA: Morgan Kaufmann.

Zhao, J., Ivan, J. N., & DeWolf, J. T. (1998). “Structural damage detection using artificial neural networks.” *Journal of Infrastructure Systems*, *4*(3), 93-101.

Zimmerman, D. C. & Kaouk, M. (1994). “Structural damage detection using a minimum rank update theory.” *Journal of Vibration and Acoustics*, *116*, 222-230.

Zimmerman, D. C. & Simmermacher, T. (1995). “Model correlation using multiple static load and vibration tests.” *AIAA Journal*, *33*(11), 2182-2188.

Zimmerman, D. C., Kaouk, M. & Simmermacher, T. (1995). “Structural damage detection using frequency response functions.” *Proceedings of 13th International Modal Analysis Conference*, 179-184.

Zimmermann, H. (1980). “OSI reference model - the ISO model of architecture for open systems interconnection.” *IEEE Trans. Communications*, *28*(4), 425-432.

List of Recent NSEL Reports