Smart Sensor Technology: A New Paradigm for Structural Health Monitoring

B. F. SPENCER, JR. AND TOMONORI NAGAYAMA

ABSTRACT:

The computational and wireless communication capabilities of smart sensors densely distributed over structures can provide rich information for structural monitoring. While smart sensor technology has seen substantial advances during recent years, interdisciplinary efforts to address issues in sensors, networks, and application specific algorithms are needed to realize their potential. This paper first discusses each of these issues, and then reports on research that combines the results to develop a structural health monitoring (SHM) system suitable for implementation on a network of smart sensors. Experimental verification is provided using Intel's Imote2 smart sensors installed on a three-dimensional truss structure. The Imote2 is employed herein because it has the high computational and wireless communication performance required for advanced SHM applications. This SHM system is then investigated from sensing, network, and SHM algorithm perspectives.

INTRODUCTION

The investment of the United States in civil infrastructure is estimated to be $20 trillion. Annual costs amount to between 8-15% of the GDP for most industrialized countries [1, 2]. This investment is likely to increase. Indeed, much attention has been focused in recent years on the declining state of the aging infrastructure in the U.S. These concerns apply not only to civil engineering structures, such as the nation's bridges, highways, and buildings, but also to other types of structures, such as the aging fleet of aircraft currently in use by domestic and foreign airlines. The ability to continuously monitor the integrity of civil infrastructure in real-time offers the opportunity to reduce maintenance and inspection costs, while providing for increased safety to the public. Furthermore, after natural disasters, it is imperative that emergency facilities and evacuation routes, including bridges and highways, be assessed for safety. Addressing all of these issues is the objective of structural health monitoring (SHM).

To efficaciously investigate damage, a dense array of sensors will be required for large civil engineering structures [3, 4]. Dense measurements can provide detailed information on civil infrastructure, which typically consists of a large number of components and has many degrees of freedom. Monitoring the Tsing Ma Bridge and

Kap Shui Mun Bridge in Hong Kong, which uses 326 channels of sensors in total and produces about 65 MB of data every hour, is an attempt toward in-depth monitoring [5]. The expense of installing traditional monitoring systems, however, has limited significantly wider-spread implementation [6–8]. For example, the total system cost, including installation, of the monitoring system on the Bill Emerson Memorial Bridge in Cape Girardeau, Missouri, USA is about $1.3M for 86 accelerometers, which makes the average installed cost per sensor a little over $15,000 dollars [9]. Costs for other bridge installations are of a similar magnitude. Smart sensors with wireless communication capability are reported to reduce installation effort to a great extent [10] and help to realize a dense array of sensors.

Though networks of densely deployed smart sensors have the potential to improve SHM dramatically, limited resources on smart sensors preclude direct application of traditional monitoring strategies on smart sensor networks. Consider the task of locating a few strategic sensors in structures such as the 2 km long Akashi-Kaikyo Bridge or the 443 m tall Sears Tower in Chicago so that these sensors can detect randomly occurring damage; such a task is intractable, if not impossible. To effectively detect arbitrary damage in structures, especially complicated structures, a dense array of sensors distributed over the entire structure will be required. However, using the traditional centralized approach for SHM will require a tremendous amount of data to be sent to such a central station. Centralized SHM strategies are not scalable to large numbers of sensors.

Noting that damage in structures is an intrinsically local phenomenon, SHM applications using smart sensors may be realized even for such structures of substantial size. Responses from sensors close to the damaged site are expected to be more heavily influenced than those remote to the damage. If data is locally processed, communication requirement will remain reasonable. Similarly, smart sensor characteristics need to be well investigated with respect to application specific requirements. Time synchronization accuracy may not be suitable for some applications. Packet loss may severely affect the performance of SHM systems. By understanding both the structure and the smart sensor network, implementable SHM systems deployed on a dense array of smart sensors can be achieved.

This paper addresses problems toward realization of SHM systems employing smart sensors and demonstrates such a system. Smart sensors are briefly reviewed first. Following description of a smart sensor system framework employed, middleware services such as data aggregation and reliable communication are developed. Finally, the SHM using smart sensors is experimentally verified by implementing the Distributed Computing Strategy (DCS) for SHM on Imote2s and monitoring a three-dimensional truss.

SMART SENSORS

The essential difference between a standard sensor and smart sensor is the latter's flexible communication and information processing capability. Each sensor has an on-board microprocessor that can be used for digital signal processing, self-diagnostics, self-identification, and self-adaptation functions. Furthermore, all smart sensor platforms have thus far employed wireless communication technology.

Some of the first efforts in developing the smart sensors for application to civil engineering structures were presented by [11–13]. Since these first efforts, numerous researchers have developed smart sensing platforms. Lynch and Loh [6] cited over 150 papers on wireless sensor networks for SHM conducted at over 50 research institutes worldwide.

Several SHM applications with smart sensors have been studied using both scale models and full-scale structure [14–20]. Sensor calibration and demonstration of data acquisition and computational capability have been performed with the ultimate goal of life-long monitoring of civil infrastructure using a dense array of smart sensors.

While smart sensor technology has seen substantial advances during recent years, interdisciplinary research efforts to address issues in sensors, networks, and application specific algorithms are needed to realize their potential. For example, accelerometers equipped with smart sensors platforms are not necessarily designed to precisely measure structural vibration in the low frequency range. Some of research gaps are summarized in Table 1.

Table 1: Research gaps.

network	**Sensor node**	**Algorithms**
Scalability	Power available	Scalability
Time synchronization	Power needed to meet performance requirements.	Distributed
Data loss	Computational speed	Data aggregation
Power efficiency	Communication bandwidth	Minimize power
Localization	Environmental hardening	Sensor fusion
adaptive network	Resolution/range	Data protocols
Fault tolerance	Sensor type	
interoperability	Digital vs analog	
middleware		

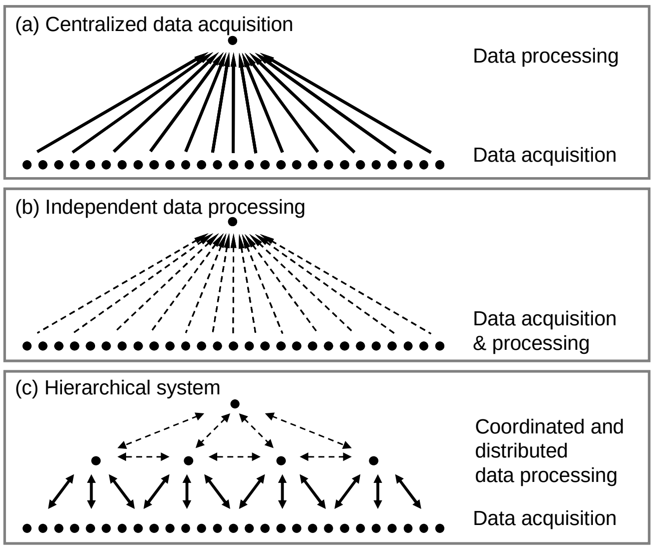
FRAMEWORK

Network topology

Most of the 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 synchronously collected for processing at a centralized location (see Figure 1a). Centralized SHM algorithms then can be applied to this data.

This approach allows for application of a wealth of traditional SHM algorithms. As the number of smart sensors increases, however, the measurement data to be centrally collected exceeds the network bandwidth, regardless of whether homerun or hopping communication is adopted. Forwarding data to the base station may take a prohibitively long time and consume a lot of power. Introducing faster communication speeds offered by nodes with ample power sources is one approach. Chintalapudi, et al. [21] utilizes a tiered approach, with lower tier nodes and powerful upper tier nodes. Assuming that the upper tier nodes have sufficient power, the limitation on the communication speed among upper tier nodes is removed; power consumption at lower tier nodes is moderate. Installing powerful nodes is sometimes impractical or can reduce the advantages of smart sensors. The limited communication bandwidth and battery power hinders the application of a centralized data acquisition approach to a large smart sensor network.


(a) Centralized data acquisition
Data processing
Data acquisition
(b) Independent data processing
Data acquisition & processing
(c) Hierarchical system
Coordinated and distributed data processing
Data acquisition
Figure 1: Smart sensor topologies.

The second group of the algorithms assumes that each smart sensor measures and processes data independently without sharing information among the neighboring nodes [10, 19, 22]. Since only the processed data is sent back to base station, communication requirements are quite modest. Consequently, this approach is scalable to a large number of smart sensors (see Figure 1b). However, the independent approach does not utilize available information from neighboring node; all spatial information is discarded. For example, mode shape information cannot be obtained. The inability to incorporate spatial information limits the effectiveness of this approach.

A hierarchical system is considered to resolve the limitations of these two approaches (see Figure 1c). Smart sensors are conceptually divided into hierarchical levels. Data processing is coordinated and distributed among sensors. The Distributed Computing Strategy for SHM (DCS) [4] is an example of such a hierarchical SHM system. Communication and data processing in DCS take place mainly in local sensor communities, reducing requirements on transmission of large amounts of data. Structural analysis of DCS takes into account measurements at multiple locations, making use of available spatial information. The DCS has the features that make it possible to be deployed on a dense array of smart sensors.

Proposed architecture

A homogeneous configuration of hardware is chosen as opposed to tiered system approaches employing resource demanding upper level nodes and less powerful lower level nodes. In addition to smart sensors, a PC is needed in this architecture as interface to users. Homogeneous configuration results in simpler programming and deployment of smart sensor nodes. Systems with homogeneous configurations can be programmed so that failure of one node does not result in failure of the systems; roles of a non-functioning node can be taken over by neighboring nodes as conceptually shown in Figure 2.

In terms of functionalities, smart sensor nodes in the proposed system are differentiated as the base station, the manager nodes, cluster heads, and leaf nodes. All the sensors deployed on a structure, in principal, work as leaf nodes. Leaf nodes receive commands from the other nodes and perform preprogrammed tasks such as sensing, data processing, and acknowledgement. The collection of leaf nodes in a neighborhood make a local sensor community. One of the nodes in each local sensor community is assigned as a cluster head and handles most of 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 cluster heads also functions as the manager sensor. When intra-cluster RF communication signals reach nodes in neighboring sensor communities, the manager deals with time sharing among sensor communities to avoid RF interference. The manager also exchanges packets directly with leaf nodes to manage operations in which all the leaf nodes participate; sensing, which is triggered by the manager sensor, is an example. The base station node is the gateway between the smart sensor network and the PC. The PC, which has the user interface, sends commands and parameters to the 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 directly with any node in 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 purpose.

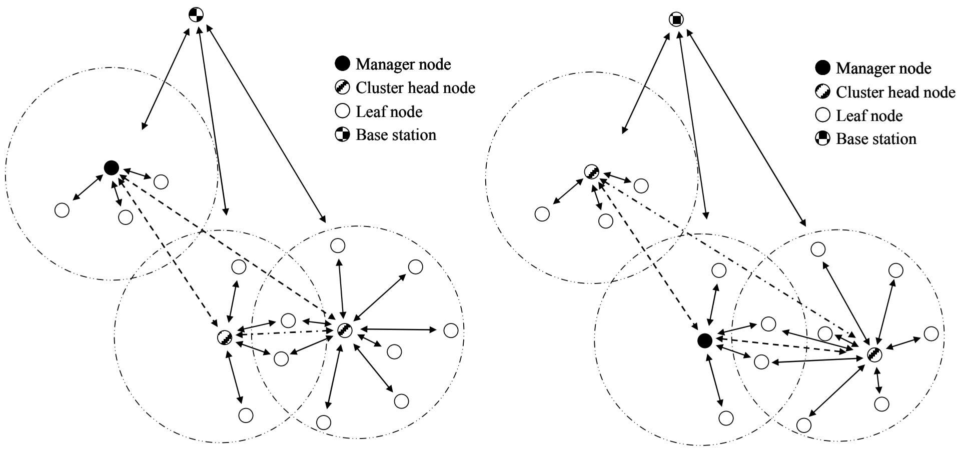


Figure 2: SHM system architecutre with interchangeable roles.

The smart sensor platform employed in this work is Intel's Imote2. The Imote2 is a new smart sensor platform developed for data intensive applications. The main board of the Imote2 incorporates a low-power XScale processor, PXA271, and an 802.15.4 radio

(ChipCon 2420). The processor speed may be scaled based on the application demands thereby improving its power usage efficiency. The Imote2 has 256 KB of SRAM, 32 MB of external SDRAM, and 32 MB of Strataflash memory. This memory is particularly important for processing the large amount of data required for dynamic monitoring of structures. Several issues regarding the efficacy of the Imote2 for SHM applications are reported by [23]

TinyOS is employed as the operating system on the Imote2. This operating system has a small memory footprint and is well-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 imposes significant limitations on SHM system functionality. A critically important issue is that 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 little control to assign priority to commands; execution timing cannot be arbitrarily controlled. This feature of TinyOS needs to be considered carefully when designing a system

MIDDLEWARE SERVICE DEVELOPMENT

In this section, the basic functionalities of smart sensors essential to SHM applications are studied and realized. Among these functionalities are data aggregation, reliable communication, and synchronized sensing. These functionalities are developed on the Imote2 platform.

Data aggregation

The amount of data transferred in SHM applications is considerable. Long vibration records will be acquired at densely distributed smart sensors. If they are collected at a single sink node using multi-hop communication, communication time easily exceeds the time necessary for any other smart sensor task. Distributed estimation of the correlation function has been proposed as a type of model-based data aggregation [24]. When the excitation can be assumed to be broadband and the structural response stationary, the correlation function between the output measurements can be used to determine modal parameters by virtue of the Natural Excitation Technique (NExT) [25]. This data aggregation method is scalable to networks of a large number of smart sensors. This data aggregation approach is described herein.

Correlation functions are, in practice, estimated from finite length records. Power and cross spectral density (PSD/CSD) functions are estimated first through the following relation [26]:

$$G_{xy}(\omega) = \frac{1}{n_d T} \sum_{i=1}^{n_d} X_i^*(\omega) Y_i(\omega) \qquad (1)$$

where $G_{xy}(\omega)$ is CSD estimation between two stationary Gaussian random process, $x(t)$ and $y(t)$. $X(\omega)$ and $Y(\omega)$ are the Fourier transform of $x(t)$ and $y(t)$; the * denotes the complex conjugate. T is time length of sample records, $x_i(t)$ and $y_i(t)$. In estimating the spectral densities, windowing of the time histories is common practice to suppress the phenomena of spectral leakage. Window functions are multiplied by the time histories, $x(t)$ and $y(t)$, prior to the Fourier transform. When $n_d = 1$, the estimate has a large random error. The random error is reduced by computing an ensemble of the estimates from n_d different or partially overlapped records. The normalized RMS error $\varepsilon\left[\left|G_{xy}(\omega)\right|\right]$ of the spectral density function estimation is given as

$$\varepsilon\left[\left|G_{xy}(\omega)\right|\right] = \frac{1}{\left|\gamma_{xy}\right|\sqrt{n_d}} \qquad (2)$$

γ_{xy} is the coherence function between $x(t)$ and $y(t)$, indicating the degree of linearity between them. Through the averaging process, the estimation error is reduced. Averaging of 10-20 times is common practice. The estimated spectral densities are then converted to correlation functions by 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 3, 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 to the reference node. The reference node calculates the spectral density. This procedure is repeated n_d times and averaged. After averaging, the inverse FFT is taken to calculate the correlation function. All the calculations take place at the reference nodes. When the spectral density is estimated from discrete time history records of length N, data to be transmitted through the radio is $N \times n_d \times (n_s - 1)$.

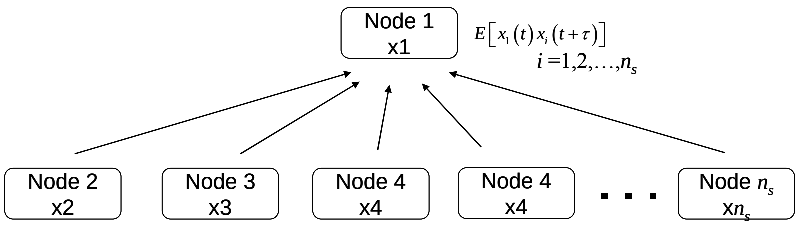


Figure 3: Centralized NExT implementation.

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 (see Figure 4). After the first measurement, the reference node broadcasts the time record to all the nodes. On receiving the record, each node calculates the cross 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

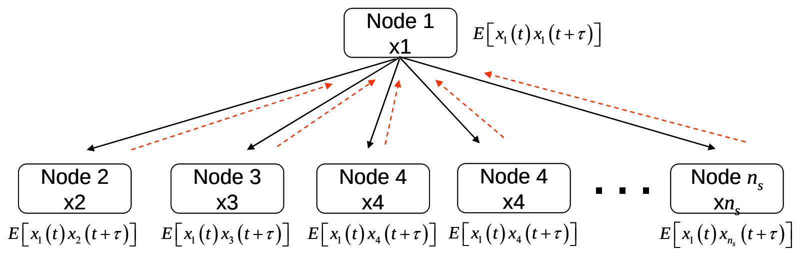


Figure 4: Distributed NExT implementation.

way, Eq. (1) is calculated on each node. Finally the inverse FFT is applied to the spectral density estimate locally. The resultant correlation function is sent back to the reference node. Because the subsequent modal analysis such as ERA uses, at most, half of the correlation function data length, $N/2$ data is sent back to the reference node from each node. The total data to be transmitted in this scheme is, therefore, $N \times n_d + N/2 \times (n_s - 1)$

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 $O\left(N \cdot \left(n_d + n_s\right)\right)$, while the first one needs to transmit to the reference sensor node data of the size of $O\left(N \cdot n_d \cdot n_s\right)$. For example, a parameter set $\{N,n_d,n_s\}=\{1024, 20,10\}$ necessitates the second approach to transfer 25,088 data points while the first one involves transmit of 184,320 data points; the reduction factor achieved by the distributed implementation is more than seven. The distributed implementation leverages knowledge regarding the application to reduce communication requirements as well as to utilize 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 application. However, Gao [4] proposed a Distributed Computing Strategy (DCS) for SHM which supports this idea. Neighboring smart sensors within single-hop communication range make local sensor communities and perform SHM in the communities. In such applications, the assumption of nodes being in single-hop range of a reference node is reasonable.

Reliable communication

RF communication is not reliable unless lost packets are specifically addressed. Packets may not be transmitted properly. When distance between nodes is too long, packets may not reach the destination. Multiple nodes trying to send packets at the same time cause packet collisions. SHM applications employing smart sensors suffer from this packet loss. 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.

When smart sensor applications come to involve more and more complicated internode data processing and are assigned more and more tasks by commands sent through packets, commands needs to be reliably delivered. Otherwise, smart sensors cannot assess the current state of neighboring nodes without intricate logic, resulting in extremely complicated programs. Reliable communication of short messages is clearly a significant help to make a SHM system with complicated internode data processing.

The need to transfer large amounts of data is not apparent. In many of SHM research attempts, data loss is not addressed. Loss of a few data points has often been considered acceptable. Some experiments luckily did not have packet loss. The effect of data loss on

SHM applications are assessed by Nagayama, et al. [27]. Packet loss is shown to degrade signals in the same way as observation noise. Reliable communication is preferable to maintain data quality. Also when loss of a large block of packets is expected, such as when devices using same frequency range pass by, resending of data is preferable.

A reliable communication protocol suitable for sending a large amount of data, as well as a protocol to send a single packet, is proposed. Each of these protocols supports multicast as well as unicast; SHM applications benefit from multicast. One example is the distributed correlation function estimation. Multicast of commands is commonly observed in smart sensor systems. Only the reliable multicast protocol for long data is briefly explained herein.

The proposed reliable communication protocol is based on a modification of Automatic Repeat reQuest (ARQ) protocol. ARQ is an error control method which repeats sending packets based on request from the receiver. On reception of packets without error, the receiver replies with positive acknowledgement (ACK). If an error is detected, the receiver sends a negative acknowledgement (NACK) and a request for retransmission. There are several ARQ protocols.

The radio component on the Imote2 is in either listening mode or in the transmission mode. During transmission, the Imote2 cannot receive packets. Without careful implementation of ARQ, the receiver may send acknowledgments while the sender is in transmission mode. Packet loss and retransmission are expected to be more frequent if transmission and reception are deeply interwoven. Scheduling interwoven transmission and reception may result in long waiting time. In the proposed protocol, the sender transmits all the packets without expecting an acknowledgement. The receiver stores all the received data in a buffer. Once the sender transmits the last packet of data, the sender repeatedly transmits a packet indicating the end of data until acknowledgement packets from the receivers are received. These acknowledgment packets also contain information about missing packets. Only missing packets are resent. At the end of transmitting missing packets, a packet requesting acknowledgment is sent again. If no receiver reports missing packets, the sender signals the end of data transfer to the receivers and itself, disengaging them from this round of reliable communication. In this way, the number of acknowledgement and retransmission can be greatly reduced (see Figure 5).

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 are usually performed in a double precision format. Even when the effective number of bits is smaller than 32, debugging of onboard data processing greatly benefits from transfer of double precision 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.

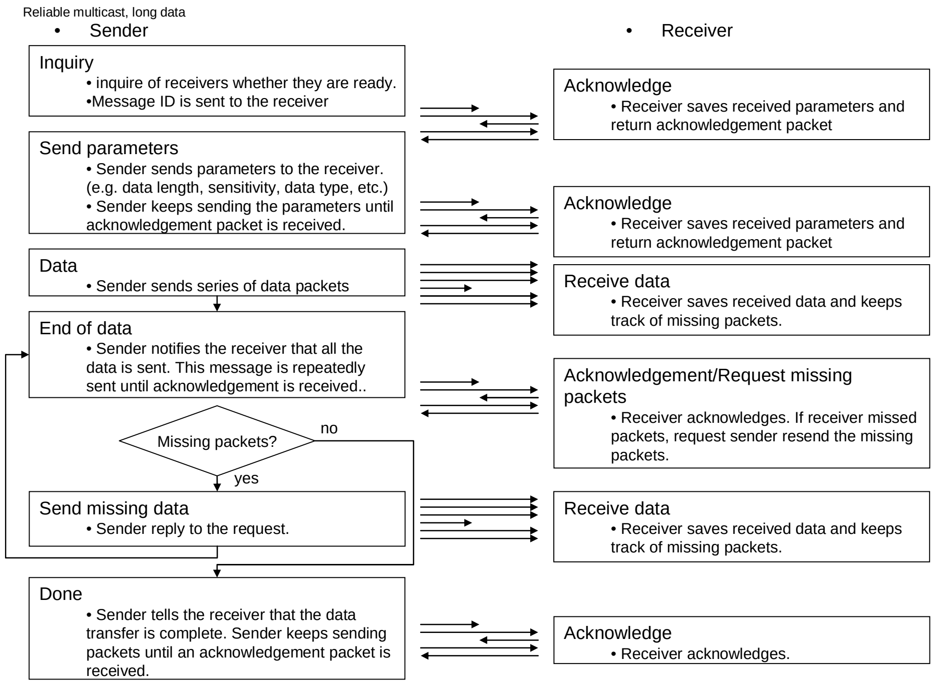

Reliable multicast, long data
• Sender
• Receiver
Inquiry
• inquire of receivers whether they are ready.
•Message ID is sent to the receiver
Acknowledge
• Receiver saves received parameters and return acknowledgement packet
Send parameters
• Sender sends parameters to the receiver. (e.g. data length, 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: Reliable communication protocol for long data records.

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 the other sensor nodes. By communicating with the surrounding nodes, smart sensors can assess relative difference among their local clocks. For example, Mica2 motes employing the Timing-sync Protocol for Sensor Network [28] 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 intensely studied, requirements on synchronization from an application perspective have not been clearly addressed. The effect of time synchronization error on SHM applications is studied by [27]. In this section, the accuracy of Flooding Time Synchronization Protocol (FTSP) [29, 30], realized on the Imote2, is evaluated for the SHM application. Time synchronization among smart sensor does not necessarily offer synchronized measurement signals. Issues toward synchronized sensing are then investigated. Finally synchronized sensing is realized utilizing resampling.

To evaluate the time synchronization error, Imote2s are programmed as follows. A beacon node transmits a beacon signal every four seconds. The others eight nodes estimate global time using the beacon packet as provided in FTSP. Two seconds after the beacon signal, the beacon node sends another packet requesting replies. The receivers get time stamps of the reception of this packet and convert them to global time stamps. The receivers take turns in reporting back these time stamps. Perfect time synchronization and time stamp of the packet reception should give the same global time stamps at all the nodes. This procedure is repeated more than 300 times. These time stamps from the eight nodes are compared with each other. Figure 6 shows the difference in the global time stamp using one of the eight nodes as a reference. The difference is generally less than 10 μsec, indicating the time synchronization error of about 10 μsec. Scattered peaks may indicate large synchronization error. However, these peaks may also be due to delays associated with global time stamping after the packet requesting replies is received.

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

While global clock estimates two seconds after sending the beacon signal are found to be accurate, local clocks drift over time. Large clock drift necessitates frequent time synchronization to maintain a certain level of accuracy.

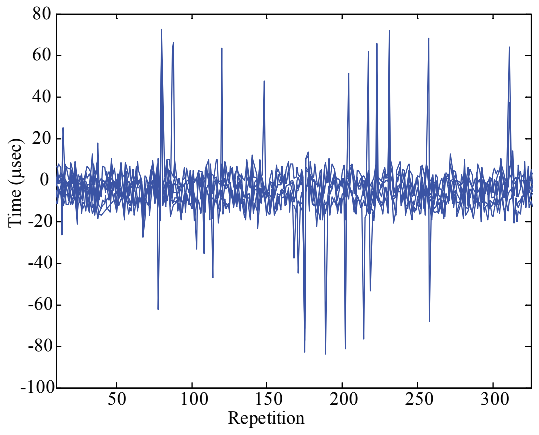


Figure 6: Time synchronization error.

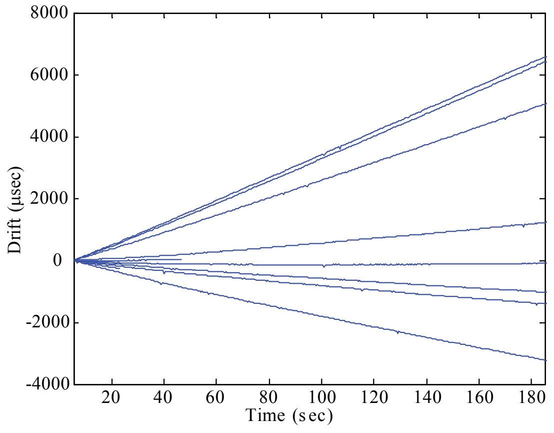


Figure 7: Drift estimation.

The same program is utilized to estimate clock drift. On reception of the packet requesting replies, the receivers get offsets of their own instead of global time stamps. The offsets are sent back to the beacon node and then to a PC. If the clocks on the nodes are ticking at exactly the same rate, the offsets should be constant over long time. This experiment, however, did not show constant offsets. Figure 7 shows the offsets of nine receiver nodes. One of them stopped responding around 45 second, exhibiting a short line on the figure. Here, the maximum clock drift among this set of Imote2 nodes is estimated to be around 50 μsec per second. This drift is small but not negligible if measurement takes a long time. For example, after 200 second measurement, time synchronization error may become as large as 10 millisecond.

One solution to address this clock drift problem is frequent time synchronization. The drawback of this approach is that time synchronization may not perform well when other tasks (e.g., sensing) are running. Time synchronization requires precise time stamping as previously explained. Sensing also requires precise timing and needs higher priority in execution. Scheduling more than one high priority tasks is challenging, especially for operating systems such as TinyOS which have no support for real-time control. If the time synchronization interval necessary to address the clock drift is shorter than sensing time length, a different solution needs to be sought.

Another approach is to compensate for the difference in clock rate. The slopes of the lines in Figure 7 approximately indicate the clock drift that needs 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 square approach.

There are several issues to be addressed toward synchronized sensing. Issues observed on the Imote2 platform include uncertainty in sensor start-up time, difference in sampling rate among sensor nodes, and fluctuation in sampling frequency over time [23]. These issues are addressed by resampling of measured time histories based on time stamps marked after a fixed number of data points are collected.

The polyphase implementation of resampling [31] by an arbitrary non-integer rational factor addresses the problem of data sampled at inaccurate frequencies. The measured time history $x[m]$ is first upsampled by about hundred zeros. Then, lowpass filter $h[i]$ is applied to yields upsampled signal $y[j]$. The outcome $z[k]$ of the resampling process is calculated by interpolating this upsampled signal.

$$\begin{aligned} z[j] &= y[p_l](p_u - p_j) + y[p_u](p_j - p_l) \\ &= \sum_{m=\lceil (p_l - N+1)/L_a \rceil}^{\lfloor p_l/L_a \rfloor} h[p_l - L_a m]x[m]\cdot(p_u - p_j) + \\ &\quad + \sum_{m=\lceil (p_u - N+1)/L_a \rceil}^{\lfloor p_u/L_a \rfloor} h[p_u - L_a m]x[m]\cdot(p_j - p_l) \qquad (3) \\ p_j &= jM_r + l_i \\ p_l &= \lfloor jM_r + l_i \rfloor \\ p_u &= p_l + 1 \end{aligned}$$

L_a is the factor of upsampling, N is the length of filter coefficients, and M_r is the non-integer factor of downsampling. $\lceil \; \rceil$ and $\lfloor \; \rfloor$ represent ceiling and floor function respectively. The initial delay, represented by l_i, is determined based on global timestamp and takes into account the inaccuracy in sampled timing. The factor, L_a / M_r, determines the rate of sampling rate conversion. The sampling rate conversion is first tested in Matlab as shown in Figure 8 and subsequently implemented on the Imote2. The resampling process implemented on Imote2 is found to yield acceleration signals with synchronization accuracy better than 50 μs.

EXAMPLE IMPLEMENTATION

The Distributed Computing Strategy (DCS) for SHM [4] is implemented on Imote2s using the developed middleware services. The SHM system realized on Imote2s is then experimentally verified using a three-dimensional truss structure.

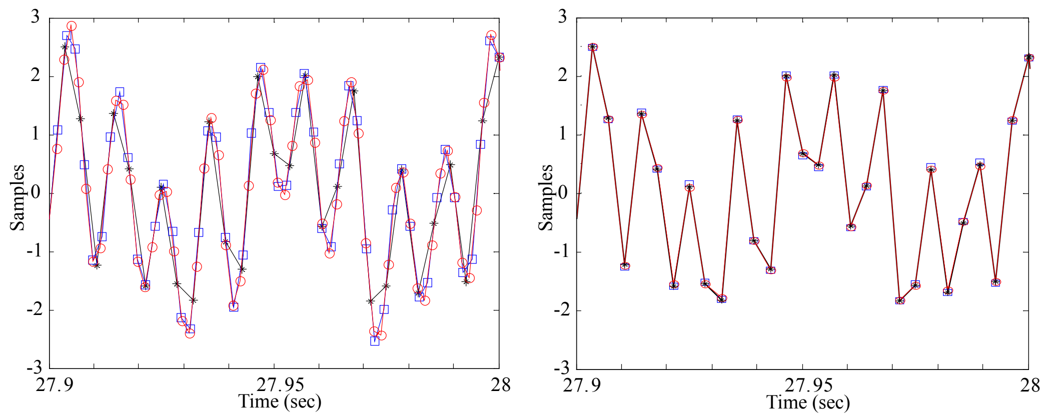


Figure 8: Signals before and after resampling.

Distributed Computing Strategy for SHM

Gao [4] proposed a Distributed Computing Strategy (DCS) for SHM that meshes well with the hierarchical architecture necessary to realize the potential of a dense network of smart sensors. This DCS approach based on the DLV method [32] does not need to centrally collect the measurement data. Instead, DCS shares data among the neighboring nodes to utilize spatial information. Due to this local data sharing with a limited number of neighboring nodes, the total amount of data to be transmitted throughout the network is kept small. Therefore, this algorithm is scalable to a large number of sensors densely deployed over large structures. While DCS does not require measurements at all the DOFs, the method improves its performance by sensing at many DOFs; DCS benefits from a dense array of smart sensors. Computer analysis and experimental validation on a simulated wireless network showed DCS is a promising SHM scheme [4]. While DCS has been shown promising as an SHM algorithm for smart sensor networks, the strategy has not been implemented on smart sensor platforms.

Gao [4] explains two methods to normalize mode shapes in DCS. One of them utilizes input force measurement while the other measures only vibration outputs under know mass perturbation. Because no Imote2 sensor board with force measurement is available and because input force of full-scale structure is difficult to measure, DCS employing mass perturbation method is chosen as the algorithm to be implemented. Bernal [33] proposed a stochastic DLV (SDLV) localization approach. This variant of the DLV method does not require normalization of mode shapes. The absence of the need for normalization greatly simplifies the SHM strategy. Therefore, the SDLV approach is also implemented on networks of Imote2s.

The DCS implementation is comprised of various steps, including numerical calculation functions, middleware services, and damage detection algorithm. Major numerical calculation functions utilized in DCS for SHM are: singular value decomposition (SVD), complex eigensolver, Fast Fourier transform, quick sort, and complex matrix inverse. These functions are either developed from scratch or functions written in C language are adapted. The performance of these functions is examined on the Imote2; execution of these functions on Imote2 yields identical outputs as those from Matlab functions with the precision of double data type. Middleware services for DCS include: data aggregation, reliable communication, and synchronized sensing. Services described earlier are adapted. The algorithm to locate damage involves NExT, ERA, and DLV. These techniques are coded as C-functions and adapted to TinyOS.

Implementation of DCS for SHM including these functions requires further consideration of the limited hardware resources. The memory space on the Imote2 is limited, and the CPU speed is slower than that of a PC. Some of numerical operations involved in DCS for SHM cannot be directly ported to Imote2s. Examples of such points needing consideration and ways of coping are as follows. Firstly, the size of the Hankel matrix utilized in ERA may exceed the available memory space on the Imote2s. The size is limited so that the matrix fits in the memory space. Application of DCS on a PC to experiment data has shown that modal parameters can be estimated accurately with a reduced-size Hankel matrix. Secondly, estimation of the stresses induced by the DLVs may involve structural analysis of the whole truss, which requires large amounts of memory and calculation time. Instead, a matrix to convert input force to stress is calculated on a PC in advance and injected to cluster heads; cluster heads needs to simply compute the product of the matrix and DLVs rather than to run the entire structural analysis. Thus, DCS can be ported to the Imote2 with careful consideration.

Imote2s are preprogrammed to autonomously accomplish the DCS. All the necessary parameters such as node ID, sensor direction, and data length are initially injected to the network from the base station. The base station sends these parameters to the manager sensor or cluster head sensors, which forward a part of the parameters to the leaf nodes. After this parameter injection, the PC connected to the base station does not need to give input to the Imote2 network. Tasks are preassigned to each command which is sent by the reliable communication protocol.

Several functionalities, which may not be necessary for complete SHM systems, are considered important for debugging purposes and are implemented as well. Measured acceleration time histories are sent back to the base station so that processing of the time history in the network can be compared to equivalent data processing on a PC. Intermediate results of DCS such as modal parameters, DLVs, and accumulated stress are also sent back to the base station for the same reason. Critical communication packets among sensor nodes are also directed to the base station using multicast reliable communication protocol so that important communication is logged at the base station for debugging. These functionalities help development and performance evaluation of the smart sensor system.

Experimental Setup

A 5.6 m long, three-dimensional truss structure at the Smart Structures Technology Laboratory (SSTL) of the University of Illinois at Urbana-Champaign (http://sst.cee.uiuc.edu/) is employed for the experimental validation (see Figure 9). The length of each bay of the truss is 0.4 m on each side. The truss sits on two rigid supports.

One end of the truss is a pinned support, and the other is a roller support. The pinned end can rotate freely with all three translations restricted. The roller end can also move in the longitudinal direction.

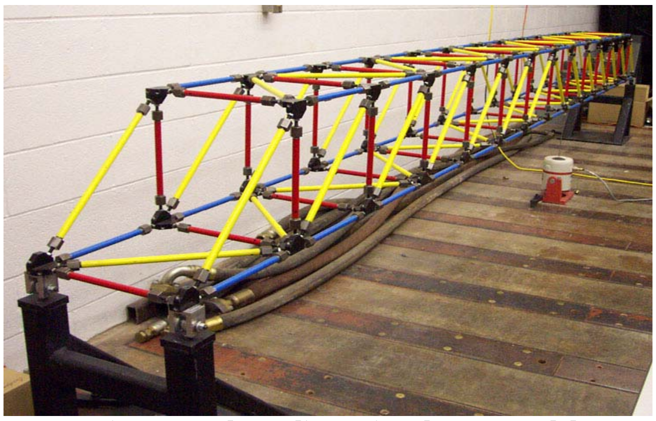


Figure 9: Three dimensional truss model.

The truss is excited vertically by a Ling Dynamic Systems permanent magnetic V408 shaker. 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 stinger.

Ten Imote2s mounted on nodes of the truss measure acceleration in three directions. The three axes of the Imote2 accelerometers are aligned with longitudinal, transverse and vertical directions. Acceleration measured at about 560 Hz is resampled to 280 Hz. Six Imote2s mounted on six front panel nodes of two consecutive bays of the truss constitutes a local sensor community and monitor structural damage within the bays. The ten Imote2s in total make three sensor communities overlapping each other.

A horizontal element on the lower cord is replaced with a thinner element to simulate damage to the truss. This replacement results in 52.7 % cross section loss of element 20 and 8 in the mass perturbation based DLV experiment and SDLV experiment respectively (see Figure 10). The cross section loss reduces stiffness in the element and is expected to be detected as damage by the DLV method.

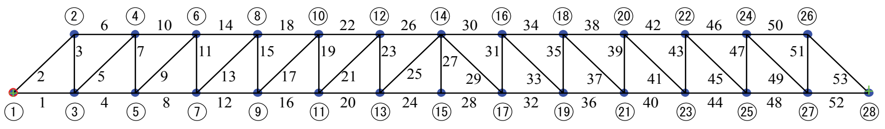


Figure 10: Node and element IDs of the truss.

Experimental Results

Through the resampling process, measured acceleration signals are synchronized to each other. The phase of the cross spectral densities indicates the synchronization accuracy. The slope of the phase indicates time synchronization error. As shown in Figure 11, the time synchronization error is approximately 30 μsec.

Modal parameters are determined from these acceleration signals. The NExT method is used to estimate the cross spectral densities and converts them to correlation functions. The Hanning window is employed in spectral density estimation. Figure 12 shows cross spectral densities between vertical acceleration signal of a cluster head node and its members' longitudinal and vertical acceleration signals. As expected, the spectral densities estimated on the Imote2s show clear peaks corresponding to structural modes. The acceleration signals are also processed on a PC to validate the numerical operation on Imote2s. The spectral densities estimated on the Imote2s in a distributed manner are found to be identical (within the numerical precision of double data type) to those calculated on the PC. Natural frequencies, damping ratios, mode shapes, EMAC, initial modal amplitude, etc. are identified from the cross spectral densities by cluster head nodes running ERA. These identified parameters are numerically the same as those identified on a PC.

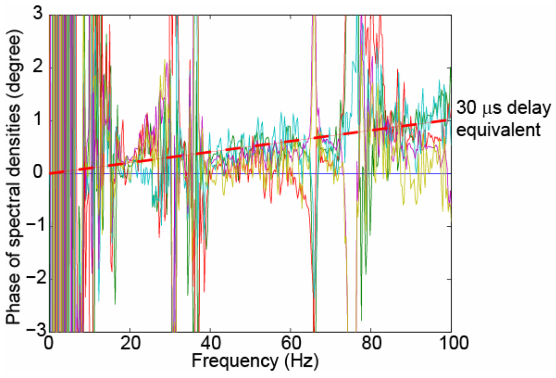


Figure 11: Phase of spectral densities.

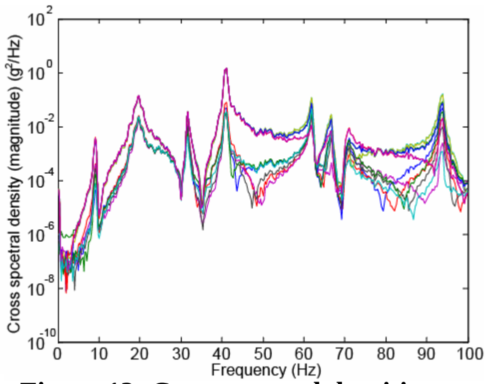


Figure 12: Cross spectral densities.

First, the mass perturbation DLV method is considered. Here, the network of Imote2s installed at node 7, 9, 11, 13, 15, 17, 19, 21, 23, and 25 (see Figure 10) estimates the mass normalization constants prior to commencement of monitoring by measuring the acceleration responses of the truss with and without known mass perturbation. Then Imote2s are installed at node 8 to 13. The modal parameters identified before and after element replacement are input to the DLV method to locate the simulated damage. The normalized accumulated stress estimated by the cluster head node, node 10, is shown in Figure 13. This local sensor community monitors element 15 to 23. The DLV method identifies damaged elements as those with small accumulated stress. Element 20, which was replaced with a thinner bar, has a normalized accumulated stress smaller than a predetermined threshold value, 0.3. While the small normalized accumulated stress successfully localized damage in this experiment, false positive/negative damage localization is sometimes observed in other cases.

The damage localization algorithm is then changed to the SDLV method and the damaged element is sought. 10 Imote2s are installed from node 2 to 11. Node 4, 6, and 8 become cluster heads and form local sensor communities around them. Each local sensor community consists of a cluster head and five surrounding nodes. Figure 14 shows the normalized accumulated stress calculated by the three adjacent cluster heads. In this experiment, element 8 is replaced with the thinner element. As shown, the Imote2s in these local sensor communities successfully detected the damaged element.

The SDLV results from the respective communities are shared among neighboring cluster heads to make judgment on damage. If neighboring nodes are consistent, the damage detection results are reported to the base station and cluster heads switches to the sleep mode. If inconsistency is observed, the neighbors retake data and apply the series of data processing again. Figure 15 shows the report from the cluster heads to the base station after applying the SDLV method and exchanging damage element information among the communities. Element 8 is identified as the only damaged element. The damage localization at the three cluster heads is consistent and the flag to indicate retaking data is set to zeros.

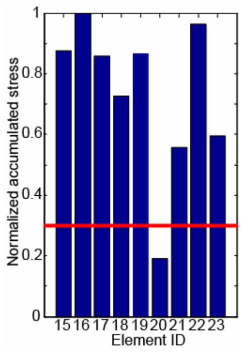


Figure 13: Normalized accumulated stress from mass perturbation DLV method.

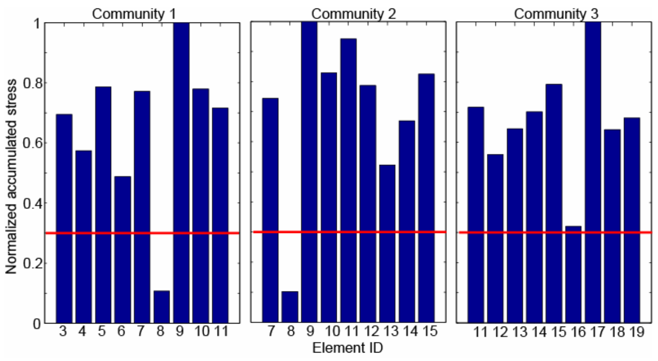
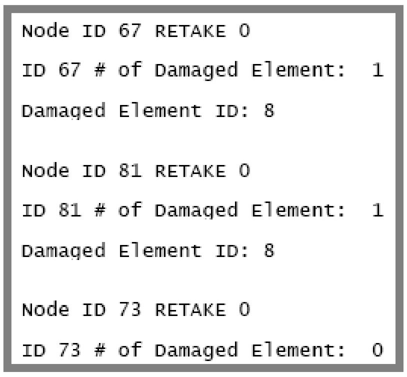


Figure 14: Normalized accumulated stress from Stochastic DLV method.



```
Node ID 67 RETAKE 0
ID 67 # of Damaged Element:  1
Damaged Element ID: 8

Node ID 81 RETAKE 0
ID 81 # of Damaged Element:  1
Damaged Element ID: 8

Node ID 73 RETAKE 0
ID 73 # of Damaged Element:  0
```

Figure 15: Report on damaged elements.

Table 2: Natural frequencies idnetified on Imote2s.

Mode	Natural frequency (Hz)
1st	**19.638**
2nd	**40.846**
3rd	**61.888**
4th	**66.714**
5th	**70.344**
6th	**93.606**

CONCLUSION

A hierarchical, distributed SHM system employing smart sensors has been developed and experimentally verified on the Imote2 platform. To realize this SHM system required various issues and algorithms to be addressed. Middleware services including data aggregation, reliable communication, and synchronized sensing are first realized on Imote2s. These services as well as numerical functions and algorithms are combined to produce the SHM system. Experimental verification using the three-dimensional truss demonstrated the efficacy of the SHM system developed herein. More details about this research can be found in Nagayama [34].

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of this research by the National Science Foundation under NSF grants CMS 03-01140 and CMS 06-00433, Dr. S.C. Liu, Program Manager. The first author would also like to acknowledge the support of a Vodafone-U.S. Foundation Graduate Fellowship.

REFERENCES

1. U.S. Census Bureau. "Value of construction Put in Place." http://www.census.gov/const/www/c30index.html. (2004)
2. Jensen, S. "Summary Outlook to 2005 for the European Construction Market." http://www.cifs.dk/scripts/artikel.asp?id=775&lng=2. (2005)
3. Spencer Jr., B.F., Ruiz-Sandoval, M.E., and Kurata, N. "Smart sensing technology: opportunities and challenges." J. of Structural Control and Health Monitoring, 11:349–368. (2004)
4. Gao., Y. Structural health monitoring strategies for smart sensor networks, Ph.D. Dissertation, University of Illinois at Urbana-Champaign, IL (2005)
5. Wong, K.-Y. "Instrumentation and health monitoring of cable-supported bridges." Structural Control and Health Monitoring, 11(2), 91-124. (2004)
6. Lynch, J. P. and Loh, K. "A summary review of wireless sensors and sensor networks for structural health monitoring." Shock and Vibration Digest, 38(2), 91-128. (2006)
7. Farrar, C. R. "Historical overview of structural health monitoring." Lecture Notes on Structural Health Monitoring using Statistical Pattern Recognition. Los Alamos Dynamics, Los Alamos, NM. (2001)
8. Celebi, M. "Seismic instrumentation of buildings (with emphasis on federal buildings)." Special GSA/USGS Project, an administrative report, United States Geological Survey, Menlo Park, CA. (2002)
9. Hartnagel, Bryan A. Personal Communication, August 22 (2006)
10. Lynch, J. P., Wang, Y., Law, K. H., Yi, J.-H., Lee, C. G., and Yun, C. B. "Validation of a large-scale wireless structural monitoring system on the Geumdang bridge." Proc., the Int. Conference on Safety and Structural Reliability, Rome, Italy. (2005)
11. Straser, E. G. and Kiremidjian A. S. "A modular visual approach to damage monitoring for civil structures." Proc., SPIE - Smart Structures and Materials, San Diego, CA, 2719, 112-122. (1996)
12. Straser, E. G. and Kiremidjian A. S. "A modular, wireless damage monitoring system for structures." The John A. Blume Earthquake Engineering Center Technical Report, 128. (1998)
13. Kiremidjian, A. S., Straser, E. G., Meng, T. H., Law, K. and Soon, H. "Structural damage monitoring for civil structures." Proc., Int. Workshop on Structural Health Monitoring, Stanford, CA, 371-382. (1997)
14. Lynch, J. P., Law, K. H., Kiremidjian, A. S., Carryer, E., Kenny, T. W., Patridge, A., and Sundararajan, A. "Validation of a wireless modular monitoring system for structures." Proc., SPIE - Smart Structures and Materials: Smart Systems for Bridges, Structures, and Highways, San Diego, CA, 4696(2), 17-21. (2002)
15. Aoki, S., Fujino, Y., and Abe, M. "Intelligent bridge maintenance system using MEMS and network technology." Proc., SPIE - Smart Systems and NDE for Civil Infrastructures, 5057, 37-42. (2003)
16. Tanner, N. A., Wait, J. R., Farrar, C. R., and Sohn, H. "Structural health monitoring using modular wireless sensors." J., Intelligent Material Systems and Structures, 14(1), 43-56. (2003)
17. Chung, H.-C., Enomoto, T., Shinozuka, M., Chou, P., Park, C., Yokoi, I., and Morishita, S. "Real time visualization of structural response with wireless MEMS sensors." Proc., 13th World Conference on Earthquake Engineering, No. 121, 1-10. (2004)

18. Nagayama, T., Ruiz-Sandoval, M., Spencer Jr., B. F., Mechitov K. A., Agha, G. “Wireless strain sensor development for civil infrastructure.” Proc., 1st Int. Workshop on Networked Sensing Systems, Tokyo, Japan, 97–100. (2004)
19. Nitta, Y., Nagayam, T., Spencer Jr., B. F., and Nishitani, A. “Rapid damage assessment for the structures utilizing smart sensor MICA2 MOTE.” Proc., 5th Int. Workshop on Structural Health Monitoring, Stanford, CA., 283-290. (2005)
20. Ou, J. P., Li, H. W., Xiao, Y. Q., Li, Q. S. “Health dynamic measurement of tall building using wireless sensor network.” Proc., SPIE - Smart Structures and Materials, 5765(1), 205-215. (2005)
21. Chintalapudi, K., Paek, J., Gnawali, O., Fu, T., Dantu, K., Caffrey, J., Govindan, R., and Johnson, E. “Structural Damage Detection and Localization Using NetSHM.” Proc 5th International Conference on Information Processing in Sensor Networks: Special track on Sensor Platform Tools and Design Methods for Networked Embedded Systems (IPSN/SPOTS'06), Nashville, TN, (2006)
22. Sohn, H., Worden, K., and Farrar, C. R. “Statistical damage classification under changing environmental and operational conditions.” J. Intelligent Material Systems and Structures, 13, 561-574. (2002)
23. Nagayama, T., Rice, J.A., and Spencer, Jr., B.F. “Efficacy of Intel’s Imote2 wireless sensor platform for structural health monitoring applications.” Proc., Asia-Pacific Workshop on Structural health Monitoring, Yokohama, Japan. (2006)
24. Nagayama, T, Spencer, B.F., Agha, G., and Mechitov, K. “Model-based Data Aggregation for Structural Monitoring Employing Smart Sensors.” 3rd International Conference on Networked Sensing Systems (INSS). (2006)
25. James, G. H., Carne, T. G., Lauffer, J. P., and Nord, A. R. “Modal testing using natural excitation.” Proc., 10th Int. Modal Analysis Conference, San Diego, CA..(1992)
26. Bendat, J. S., and Piersol, A.G. Random data: analysis and measurement procedures, John Wiley and Sons, Inc. (2000)
27. Nagayama, T. Sim, S.H., Miyamori, Y., Spencer Jr., B.F. “Issues in Structural Health Monitoring Employing Smart Sensors.” Smart Structures and Systems (in review)
28. Ganeriwal, S., Kumar, R., and Srivastava, M. B. “Timing-sync protocol for sensor networks.” Proc., 1st International Conference On Embedded Networked Sensor Systems, Los Angeles, CA, 138 - 149. (2003)
29. Maroti, M. Kusy, B., Simon, G., and Ledeczi, A. “The flooding time synchronization protocol.” Proc., 2nd International Conference On Embedded Networked Sensor Systems, Baltimore, MD, 39-49. (2004)
30. Mechitov, K., Kim, W., Agha, G., and Nagayama, T. “High-frequency distributed sensing for structure monitoring.” Proc., 1st Int. Workshop on Networked Sensing Systems, Tokyo, Japan, 101–105.(2004)
31. Oppenheim, A.V., Schafer, R.W., Buck, J.R. “Discrete-Time Signal Processing.” Prentice Hall. (1999)
32. Bernal, D. “Load vectors for damage localization.” J. of Engineering Mechanics, 128(1), 7-14. (2002)
33. Bernal, D. “Flexibility-based damage localization from stochastic realization results.” J of Engineering Mechanics, 132(6), 651-658. (2006)
34. Nagayama, T. Structural health monitoring using smart sensors, Ph.D. Dissertation, University of Illinois at Urbana-Champaign, IL (2007)