Structural Damage Detection and Localization Using NetSHM

Krishna Chintalapudi
krishna.chintalapudi@rtc.bosch.com

Jeongyeup Paek
jpaek@enl.usc.edu

Omprakash Gnawali
gnawali@usc.edu

Tat S. Fu
tsf@usc.edu

Karthik Dantu
dantu@usc.edu

John Caffrey
jcaffrey@usc.edu

Ramesh Govindan
ramesh@usc.edu

Erik Johnson
ejohnson@usc.edu

Sami Masri
masri@usc.edu

University of Southern California
Los Angeles, CA 90089.

ABSTRACT

Structural health monitoring (SHM) is an important application area for wireless sensor networks. SHM techniques attempt to autonomously detect and localize damage in large civil structures. Structural engineers often implement and test SHM algorithms in a higher level language such as C/Matlab. In this paper, we describe the design and evaluation of NetSHM, a sensor network system that allows structural engineers to program SHM applications in Matlab or C at a high level of abstraction. In particular, structural engineers do not have to understand the intricacies of wireless networking, or the details of sensor data acquisition. We have implemented a damage detection technique and a damage localization technique on a complete NetSHM prototype. Our experiments on small and medium-scale structures show that NetSHM is able to detect and localized damage perfectly with very few false-positives and no false negatives, and that it is robust even in realistic wireless environments.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation

Keywords

Wireless sensor networks, structural health monitoring

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

1. INTRODUCTION

Structural health monitoring (SHM) [1] is a vast, interdisciplinary area of research whose literature spans several decades. Wireless sensor networks promise cheap and dense instrumentation for structural monitoring [2]. Recent work has demonstrated the feasibility of continuous structural data collection using a wireless network [3, 4, 5]. This paper takes the next step in the evolution of this research, examining how wireless sensor networks might be used for more sophisticated structural monitoring tasks.

The central focus of SHM research is the detection and localization of damage in a variety of structures [6]. Broadly speaking, SHM techniques for detecting [7, 8, 9] and localizing [10] damage (henceforth, SHM techniques) rely on measuring structural response to ambient vibrations or forced excitation. Ambient vibrations can be caused by earthquakes, wind, or passing vehicles, and forced vibrations can be delivered by hydraulic or piezoelectric shakers. A variety of sensors, such as accelerometers, strain gauges, and displacement sensors can be used to measure structural response. SHM techniques infer the existence and location of damage by detecting differences in local or global structural response before and after damage.

Wireless sensor networks have an important role to play in SHM. Wireless sensors simplify the deployment of instrumentation, and can greatly reduce cabling costs [11]. Furthermore, a dense deployment of wireless sensors can increase the accuracy of SHM techniques. This follows from the fact that many SHM techniques measure properties of a structure's *modes*. A mode is the spatio-temporal deformation pattern exhibited by a structure in response to a vibration. Damage can alter one or more modes. Large structures can have several hundred modes, and the larger the number of sensors, the larger the likelihood of accurately capturing all the modes. On the other hand, SHM presents many challenges for wireless sensor networks [12]: SHM applications need reliable high data-rate delivery, and fine time-synchronization.

For wireless sensor networks to be adopted for SHM, a key enabler will be a system that allows structural engineers to program SHM techniques in an idiom they are comfort-

able with. This paper discusses the design of NetSHM, a *programmable*, *re-usable* and *evolvable* software system for implementing SHM techniques on wireless sensor-actuator networks. Structural engineers write SHM applications in Matlab or C at a high level of abstraction; their programs explicitly *task* collections of sensors and actuators at specified times, then process the retrieved data using data manipulation constructs native to the corresponding programming language. In NetSHM, sensors can also be tasked to locally process raw sensor data for energy-efficiency. At this high level of abstraction, structural engineers need not be exposed to the intricacies of wireless networking and routing, nor to the platform specific details of sensor data acquisition.

The NetSHM system architecture (Section 2) is based on the observation that, to deal with the data rate requirements of SHM applications [4], wireless sensor-actuator networks for structural monitoring will be hierarchical. These networks will consists of two tiers: the lower-tier comprised of mote-class wireless sensor nodes enable flexible deployment on a structure, and an upper-tier comprised of higher-capacity nodes (either PCs or Stargate-class nodes) that provide the bandwidth scaling. NetSHM *leverages this hierarchical network to realize a novel functional decomposition between the two tiers.* In NetSHM, all applications run on upper-tier nodes and *task* individual motes to collect, and possibly process, data. Motes transmit the raw or processed data to the relevant upper-tier node, perhaps over multiple hops. Such an architecture permits re-use of the lower-tier systems components, and a flexibly designed tasking interface can let applications use increased processing power on the motes as the technology evolves.

Our NetSHM prototype (Section 2) is a fairly complete realization of this architecture. We have implemented a routing subsystem that enables communication between the two tiers, and a reliable delivery mechanism for conveying tasks and results between the two layers. We have also implemented a tasking interface that permits triggered data collection from the lower-tier.

On our NetSHM prototype, we have implemented two qualitatively different SHM applications, one which detects damage using shifts in modal frequencies, and another which localizes damage based on mode shape changes. In both these applications, data acquisition is *triggered* by forced vibrations on the structure; this mode of operation is suitable for wireless sensor networks since nodes can sleep between tests to conserve energy. Furthermore, some of our SHM techniques are amenable to *local processing*, where raw sensor data is processed before being transmitted to the base station, further conserving energy. We show how a previously proposed localization technique can be adapted to take advantage of local processing.

We have extensively (Section 3) evaluated these applications on a scaled model of a 4-story building. We find that our applications are able to detect and localize *all* the damage patterns we studied. We also find that these applications are amenable to a highly energy-efficient implementation. We have also evaluated a larger deployment of a NetSHM prototype on a full scale imitation hospital ceiling. We find that NetSHM can provide robust, low-latency triggered data acquisition even in realistic wireless environments. Finally, we know of no prior work that has examined a software system for SHM application development. Closest in spirit to our work is that of Kottapalli *et al.* [13] who also sketch a two-tier architecture for SHM applications, but focus on hardware and MAC layer designs for the two tiers.

2. NetSHM ARCHITECTURE, DESIGN AND IMPLEMENTATION

In this section we describe the architecture, design and implementation of our NetSHM prototype.

2.1 Goals

In addition to the data rate, reliable delivery and time synchronization requirements imposed by SHM applications, we set three design goals for NetSHM. (In what follows, we refer to an SHM technique implemented on NetSHM as an *SHM application.*)

First, NetSHM should present a programming abstraction familiar to structural engineers. In particular, the NetSHM programmer should not be exposed to the intricacies of wireless communication, energy management, and device resource constraints. Without this requirement, we see little likelihood of sensor networks being adopted in this application domain.

Second, NetSHM must be designed to be re-usable in two distinct senses of the term. System components should not have to be re-designed or re-implemented for different SHM applications. For instance, one should not have to write a new routing or time synchronization protocol when implementing a new SHM application on NetSHM. This is a very difficult objective since the space of SHM techniques is large.

Finally, NetSHM applications should not have to be re-implemented as technology evolves. The mote-class devices today cannot support some of the signal processing tasks that SHM applications demand; given their memory constraints, it is possible to implement Fast Fourier Transforms on the motes, but just barely so. However, as these devices evolve to have more on-board processing and memory, the NetSHM system should be able to transparently make use of these resources in order to increase system lifetime.

2.2 The NetSHM Architecture

NetSHM employs a different, yet simple and intuitive architecture for two-tiered sensor/actuator networks (Figure 1). In NetSHM, SHM applications run on an upper-tier node (*e.g.,* a PC). An application may run on any node. Restricting programs to run on upper-tier nodes is motivated by the signal processing needs of SHM applications. As technology advances, upper-tier nodes will always have far more processing and memory than lower-tier nodes (motes) and SHM applications can leverage these resources to improve the quality of detection and localization.

In NetSHM, an SHM application can address a message to a specific mote. This individual mote addressability contradicts what previous research has assumed, but more naturally captures the way structural engineers think about SHM techniques. Specifically, their descriptions of algorithms employ spatial concepts (like mode shapes), and being able to specify individual sensors or sensing points is a useful capability that helps them program SHM techniques. Individual mote addressability enables SHM applications to selectively address individual shakers, or to address a subset of motes in an over-engineered deployment.

SHM applications can send messages to individual motes, but they cannot perform arbitrary computations on them.

In the NetSHM architecture, an application is restricted to *tasking* a mote in one of four ways: 1. Collect raw sensor data within a specified time interval and with specified sensor parameters (*e.g.,* sampling rate), and transmit it back to the upper-tier node running the application; 2. Collect raw sensor data within a specified time interval and with specified sensor parameters (*e.g.,* sampling rate), but locally process the raw samples and transmit the processed data to the upper-tier node running the application. At any given instant, a mote exports a pre-defined library of processing functions. This library can evolve over time. This library can contain functions ranging from simple windowed averaging and thresholding, to more complicated computation of FFTs and ARMA coefficients; 3. Actuate an attached shaker using a specified type of excitation at a specified time, or according to a specified schedule; and, 4. Get and set device parameters and statistics. This can be used for monitoring and management of the lower-tier devices.

2.3 The Programming Interface

One explicit goal of NetSHM is to raise the level of abstraction for programming SHM applications to the point where structural engineers should not have to understand the intricacies of sensor networking. Without this, we believe it is unlikely that sensor networks will be used for SHM. To guide our choice of the NetSHM programming interface, we look to current practice in the structural engineering community. Many structural engineers extensively simulate models of structures in software before prototyping or building them. Matlab is often used for analytical models, and NASTRAN (implemented in C) for finite-element modeling.

Our NetSHM prototype provides programmers with a suite of Matlab and C functions that they can use to implement SHM applications. Our programming interface closely follows the tasking interface described in Section 2.2. A triggered SHM application implemented using this interface first creates one or more logical groups of motes that need to participate in the application. This group construct is a convenient abstraction for addressing collections of lower-tier nodes. The application then tasks a group to start collecting vibration data at a specified relative time with a specified sampling rate. It also tasks a group of shakers to excite the structure at a specified time. The results of this test are returned asynchronously to the SHM application, and can be manipulated by the application as a Matlab dataset. Figure 6 shows an example of a damage detection application written using this interface. (In the code snippets, functions beginning with "NetSHM" are part of the interface. We have omitted a detailed description of the NetSHM API for brevity.) We describe this example in greater detail in a later section.

2.4 The Software Structure

NetSHM consists of two distinct stacks, one for the upper-tier nodes (Figure 2), and one for the motes (Figure 3).

At the top of the upper-tier stack, the NetSHM programming interface is built on top of a task library. In our current prototype, the task library is implemented in C, and the NetSHM Matlab interface functions make calls to the library via Matlab Mex function wrappers. The library translates the interface functions into a sequence of tasks that are transmitted to motes. It invokes a reliability mechanism implemented on top of a robust routing layer. These two components are described below.

The task library itself is conceptually simple. Interface calls to create node groups create local state in the library that associate a group identifier with a set of nodes. In principle, the library would interface with a resource discovery component for determining the identity, location, and other characteristics of the motes deployed on a structure. In our current prototype, this information is manually configured. This is a reasonable short-term design since placing sensors on structures typically involves a great deal of planning and human intervention. Longer-term, of course, automated node discovery mechanisms can simplify deployment. Interface calls to command a group of nodes to collect sensor data, or to excite the structure are translated by the task library into messages that are individually sent to the relevant motes. Our current stack does not export a multicast delivery abstraction, and we have left this to future work.

The mote stack is similarly implemented. Tasks from SHM applications are delivered using the routing layer to the addressed mote. At each intermediate hop, network packets may be retransmitted to improve the likelihood of delivery. At the destination, the reliability layer implements end-to-end error recovery and sequenced delivery to the tasking layer. This latter layer interprets the tasking commands and executes them appropriately. Executing a task may involve activating a sensor or a shaker at a specified time, and then processing raw sensor data. Results from a task are delivered reliably back to the corresponding upper-tier node. In our current prototype, we have not implemented any local processing, since even the most basic SHM processing (an FFT on vibration samples) is beyond the reach of the platform NetSHM currently executes on (the Mica-Z). In our experience, the memory constraint on this platforms inhibits this functionality, and we believe that such local processing can be implemented on the next generation of sensor platforms such as the Tmote Sky or the Intel Research motes.

NetSHM uses the FTSP [14] implementation in the TinyOS tree for time synchronization, with some minor modifications for compatibility. We also changed the default clock frequency so that the clock wrap-around time is sufficiently large, yet bounds time synchronization error to within a few hundred microseconds, well within one sample time (vibrations from large structures are usually sampled at hundreds of Hz).

The current NetSHM prototype lacks two important components: support for the execution of multiple concurrent SHM applications, and support for duty cycling. These components require substantial research and are beyond the scope of this work. Concurrent application execution will allow users, for example, to improve the accuracy of damage localization by concurrently running two qualitatively different damage localization algorithms. Support for concurrent execution can be implemented by a distributed resource management layer at the upper-tier, which knows about and arbitrates between overlapping tasks issued to one or more motes. Support for network-wide duty-cycling can leverage prior work on the design of long-lived sensor networks but needs to support unpredictable task arrivals.

2.5 Routing

A core component of NetSHM is the module that sup-

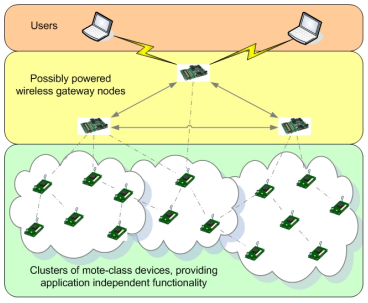

Figure 1: The organizational hierarchy of NetSHM

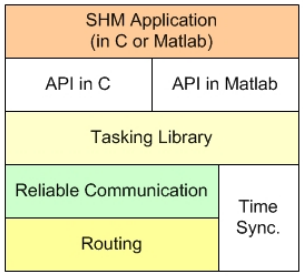

Figure 2: NetSHM Stack on the gateway nodes

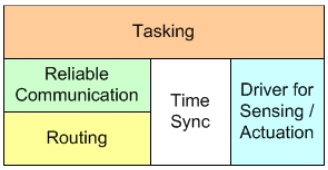

Figure 3: NetSHM stack on the mote class nodes

ports robust dynamic routing. Unlike traditional communication networks, NetSHM does not require any-to-any routing. Rather, upper-tier nodes need to be able to communicate with any mote, and vice-versa. A mote in NetSHM never originates a message destined to another mote. Upper-tier nodes can communicate with each other; in NetSHM, an application may be distributed across many upper-tier nodes (although our current applications are not). The requirements for robustness in NetSHM are, however, the same as in other networks: as long as there exists a communication path between an upper-tier node and a mote, the two of them should be able to exchange packets.

All NetSHM nodes (upper or lower tier) derive their addresses from the same flat address space. In our implementation, lower tier nodes use the TinyOS node IDs while the upper tier nodes are manually configured. Admittedly, flat addressing does not scale beyond a few hundred nodes and devising scalable, auto-configurable addressing is a key challenge for NetSHM.

Any-to-any routing on the upper-tier can potentially borrow solutions from the ad-hoc routing literature. For expediency, we chose to implement a simple distance-vector type scheme on the upper-tier. This routing protocol not only disseminates routes to upper-tier nodes, but also distributes routes to the motes imported from the gateways (described below). Routes to motes are tagged with the address of the gateway, enabling upper-tier nodes to reach any mote. This design is conceptually similar to the kind of hierarchical routing employed in the Internet.

For routing from motes to an upper-tier gateway, our prototype uses code from the CENS Extensible Sensing System. This system constructs several routing trees in the lower tier, one tree rooted at each gateway. Each mote chooses the "best" gateway, using a cost metric similar to [15].

For routing from an upper-tier gateway to the motes, we have implemented a simple flat routing protocol built on this forest of trees. In our protocol, each node propagates, to its parent, reachability to all nodes within the subtree rooted at itself. This is achieved using a periodic routing table transmission to the parent. Eventually, the gateway receives routes to all motes within its subtree, and it then exports these routes into the upper-tier any-to-any routing protocol as described above.

2.6 Reliable Delivery

NetSHM provides reliable delivery of tasks from upper-tier nodes to motes, and raw samples or processed data from motes back to the corresponding upper-tier node. In NetSHM, reliable delivery is *transactional* in the sense that data is always sent in response to a task. However, these transactions can be *asynchronous*. The response to a task can be received well after that task is issued by the SHM application. For example, a task might require a sensor to respond at a certain time, or in response to a significant external event. Furthermore, these transactions can be *asymmetric*, since task descriptions are concise, but responses to tasks can transfer significantly more data. These two differences motivate different reliable transfer abstractions and provide opportunities for optimized implementations relative to existing reliable delivery mechanisms such as TCP.

Rather than provide a single reliable delivery abstraction between the two tiers, NetSHM leverages the asymmetry in the underlying architecture to provide two different reliable delivery abstractions: reliable packet delivery from the upper-tier nodes to motes (for tasks), and reliable and sequenced stream delivery from the motes to the upper-tier (for data).

Because tasks are smaller and fit in one packet, the implementation of the packet delivery abstraction can be optimized in two ways. First, packet delivery avoids the overhead of connection establishment and teardown, relying instead on a simple end-to-end acknowledgment. Second, this more specialized abstraction avoids the need to provision large buffers for sequencing data on the resource-constrained motes.

The stream delivery abstraction is conceptually very similar to TCP, but the implementation details are subtly different. The stream delivery abstraction is implemented using negative end-to-end acknowledgements instead of a cumulative acknowledgement as in TCP. This permits faster recovery of multiple losses. In our current implementation, retransmission buffers at the sending mote are stored in its EEPROM. For improved performance, stream delivery relies on a limited number of hop-by-hop retransmissions. The receiving end ensures sequenced delivery to the application by buffering packets. Our current implementation does not incorporate any congestion control mechanisms, but rate limits the sender to a configurable transmission rate. We have left congestion adaptation to future work.

Stream delivery uses a connection establishment mechanism very similar to that of TCP. However, because stream delivery is fundamentally simplex, the connection establishment state diagram is slightly simpler than that of TCP (omitted for brevity) and requires fewer handshakes for connection establishment and teardown.

Finally, both stream and packet delivery work transpar-

ently across the two tiers in NetSHM. Both types of reliable delivery can traverse multiple hops on both tiers of the network but there is almost no functional difference between our implementations for the two layers, with one exception: between neighboring motes our implementation uses MAC-layers ACKs to perform hop-by-hop retransmissions, but between two upper-tier neighbors, we use a TCP "tunnel" for implementation convenience.

3. EXPERIMENTAL RESULTS

We have implemented a modal frequency shift based damage detection technique, and a mode-shape estimation based damage localization technique on our NetSHM prototype. We have tested these algorithms on a 48-inch scale model of a 4-story building. We have also implemented a triggered data collection application on a full-scale seismic test structure, to get a sense of NetSHM performance under realistic wireless conditions. This section describes these experiments.

3.1 Experimental Setup

The *scaled building model* (Figure 5) is 48 inches high, with 1/2x12x18-inch aluminum plates which serve as floors and are supported by 1/2x1/8-inch steel columns. Removable 5.5 lbs/inch springs serve as braces between the floors of the structure. These strings augment the stiffness between the floors. Damage is induced by removing these springs from the structure. The building has four wirelessly controlled shakers built using off-the-shelf components. These can be tasked via an attached Mica-Z mote to deliver impulses to the top floor of the structure.

The *seismic test structure* (Figure 4) is a platform for conducting seismic experiments on a full-scale realistic imitation of a 28'×48' hospital ceiling. The ceiling is complete with functional electric lights, fire sprinklers, drop ceiling installations and water pipes carrying water. Furthermore, the ceiling is designed to support 10,000 lb of weight. The entire ceiling can be subjected to uni-axial motion with a peak-to-peak stroke of 10 inches, using a 55,000 lb MTS hydraulic actuator having a ±5 inch stroke. The hydraulic pump delivers up to 40 GPM at 3000 PSI. The total weight of the moving portion of the test structure is approximately 12,000 lb. Currently, the test structure requires a human-in-the-loop to actuate the shaker.

Our NetSHM prototype runs on a hierarchical network of PCs, Stargates and Mica-Z motes. We later describe the detailed network configuration for each of our two structures.

Attached to the Mica-Z motes is a *vibration card* specially designed for high-quality vibration sensing. The vibration card can be programmed to sample at frequencies from 5Hz to 20KHz at 16 bits per sample and has a programmable anti-aliasing filter to accommodate different sampling rates. The 16-bit ADC of the vibration card is controlled by an on-board microprocessor, which in turn can be commanded by the attached Mica-Z mote via a serial port. The stored samples can be retrieved in one shot from the on-card 64K byte SRAM by issuing commands over the serial port. We modified the card firmware to support retrieval of blocks of samples from the card's RAM. This enabled us to conserve memory on the Mica-Z. Finally, we attached highly sensitive tri-axial accelerometers (dynamic range of -2.5g-2.5g, sensitivity in the μg range), to the vibration card.

```
function k = getStiffnessFromBuilding()
 %create a group for sensors
idSensors = NetSHMCreateGroup([16,7,13,14,5,2,4,3]);
 %create a group for actuators
 idActuator = NetSHMCreateGroup([6]);
 %actuate after 22 seconds
 NetSHMCmdActuate(gidActuator,22);
%task motes to sense and send data
% 4000 samples at 200Hz along x axis   starting 20 secs from now
samples = NetSHMGetSamples(gidSensors,20,200,1,4) ;
%perform ERA on the samples
[sys,pole,modeShapes] = era(samples,1/200);
 %find  eigen values and vectors of A
 % for the correspondign continous time system
 V = log(pole(1:2:end))*200;
 U = sys.c*modeShapes(:,1:2:end)
 %find the least mean square estimate
 k = LMSSolution(V,U);
```

Figure 8: Code for the damage localization application

3.2 NetSHM Applications for Damage Detection and Localization

We have implemented two SHM techniques, one for damage detection and another for damage location, as NetSHM applications. The damage detection technique is an instance of a modal frequency shift based approach, and the localization algorithm is borrowed from the structural engineering literature. We emphasize that it is not our intent to devise novel SHM techniques. Rather, our intent is to show how to take existing SHM techniques, modify them to leverage local computation, and to understand the ease of programming them using our NetSHM prototype.

Damage Detection. We have implemented a simple modal frequency shift based damage detection application [16] in NetSHM. The code for this application is shown in Figure 6. This application works as follows: 1. Actuate the structure and collect the structural response histories from all the sensors; 2. Estimate the power spectral densities for each of the collected structural response histories–we do this by first estimating the auto-correlation of each history followed by a Fourier transform of the autocorrelations, and the magnitude of the Fourier transform provides the power spectral density; 3. Perform peak detection on the power spectral densities of the time histories and select dominant peaks (greater than a certain threshold of energy) to create a set of modal frequencies for each structural response history; and, 4. The complete set of the modal frequencies is the union of the modal frequencies discovered for each structural response history.

In NetSHM, such a technique can be implemented by tasking the actuators and sensors in Step 1, then performing Steps 2 through 4 on an upper-tier node. When mote technology evolves to the point where it becomes feasible to compute peaks in power spectral densities at the motes, NetSHM can (transparently to the application) push Steps 2 and 3 to be computed locally on the motes. Each mote would then transmit the modal frequency peaks alone, resulting in significant communication energy savings. We quantify the extent of this savings in a later section.

Figure 4: The seismic test structure

Figure 5: The 48 inch scaled model of a 4-story building

```
function shift = getModalShiftsFromBuilding()
 %create a group for sensors
idSensors = NetSHMCreateGroup([16,7,13,14,5,2,4,3]);
 %create a group for actuators
 idActuator = NetSHMCreateGroup([6]);
 %actuate after 22 seconds
 NetSHMCmdActuate(gidActuator,22);
%task motes to sense and send data
% 4000 samples at 200Hz along x axis   starting 20 secs from now
samples = NetSHMGetSamples(gidSensors,20,200,1,4) ;
 % find the modal frequencies from all the samples
 modes = findModes(samples);
 %read the originally stored modes
 load originalModes ;
 %detect possible damage
 shift = findModalFreqShifts(modes,originalModes);
```

Figure 6: Code in matlab for detecting damage in the structure

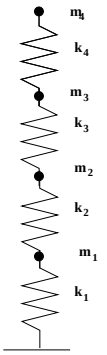

Figure 7: The lumped mass spring model of the building

Damage Localization. The damage localization application that we have implemented in NETSHM is that of Caicedo *et al.* [17]. This technique is specifically designed for multi-storied buildings and can localize the floors where the damage has occurred. The scheme models a multi-storied building as a mass spring system (Figure 7). The stiffness k_i of each spring is the combined stiffness of all the members of the i^{th} floor and the mass m_i is the entire mass of each floor. Based on the responses to vibration, our localization algorithm computes the *change in stiffness* on each floor, inferring that damage has occurred on a floor whose stiffness changes significantly.

For our 4-story building model, the solution to finding the stiffness of each floor is given by:

$$\psi_j = \mathbf{k}\mu_j \quad (1)$$

$$\psi_j = \begin{bmatrix} v_{j1} & v_{j1}-v_{j2} & 0 & 0 \\ 0 & v_{j2}-v_{j1} & v_{j2}-v_{j3} & 0 \\ 0 & 0 & v_{j3}-v_{j2} & v_{j3}-v_{j4} \\ 0 & 0 & 0 & v_{j4}-v_{j3} \end{bmatrix} \quad (2)$$

$$\mathbf{k} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}^T \quad (3)$$

$$\mu_j = \lambda_j \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \end{bmatrix} \quad (4)$$

Here, $\mathbf{v}_j = \begin{bmatrix} v_{j1} & v_{j2} & v_{j3} & v_{j4} \end{bmatrix}^T$ is the j^{th} mode shape induced in the structure and λ_j is the corresponding eigenvalue. This leads to the following system of equations:

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \phi_l \end{bmatrix} = \mathbf{k} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_l \end{bmatrix} \quad (5)$$

The stiffness is estimated as a least-mean-square fit solution over all dominant modes in the structure. To estimate the mode shapes we used the ERA algorithm described in [17].

For brevity, we omit a detailed discussion of the application, but this computation is fundamentally centralized. As such, this localization technique can be implemented by tasking sensors and actuators in a manner similar to Step 1 of the damage detection implementation, and estimating the change in stiffnesses at an upper-tier node. The NETSHM code for this application is shown in Figure 8.

3.3 Detection and Localization Results

We deployed and tested these SHM applications on the scaled building model. On this model, we deployed 8 Mica-Z modes with vibration cards, and another Mica-Z mote attached to our actuators. Two motes were deployed on each floor of the building and each mote was attached to a tri-axial accelerometer. This configuration is necessary since, for small deformations, each floor has three degrees of freedom (x, y and θ), and at least 2 accelerometers are necessary to estimate these quantities. In our setup, accelerometers were placed at two diagonally opposite corners of each floor and sample the structure at 200Hz.

The upper-tier network was formed by a PC and a Stargate, the latter acting as a single gateway to the motes (we discuss experiments with multiple gateways in Section 3.4). The PC ran our NETSHM applications and each application was programmed to run a single test. In each application, sensors were tasked to begin sampling at a specified time, and the actuators were tasked to deliver an impulse about 2 seconds after that. Each sensor collected the structural response for 30 seconds, then transmitted the data to the PC using our reliable transmission and routing modules.

We report results obtained by running our NETSHM applications on nine different damage scenarios, listed in Table 1. For each application, we ran each scenario five times, and present below the aggregated results from these tests.

Damage Detection Results. In Table 1, second column, we list all the estimated modal frequencies encountered for each configuration. The spectral resolution of our scheme was 0.1Hz in our scheme and we found the estimated modal frequencies to be consistent across all the trials for every damaged case.

Damage in a structure may result in a decrease in modal energy at some frequencies, accentuate an originally dormant mode and make it appear, or even make a mode disappear entirely. Typically the higher the frequency of the mode, the higher the shift in frequencies after damage. As seen in Table 1 the modal frequencies that exhibit a change from the common damage case *i.e.,* Case 0 are depicted in bold. *Our implementation is able to detect changes at least in one mode for every test case.* This is a very encouraging result which validates that, at least for test structures,

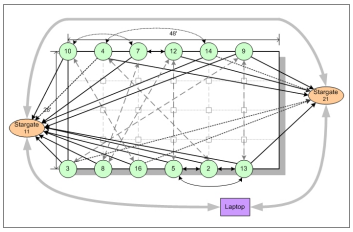

Figure 9: Layout of the motes in the seismic structure

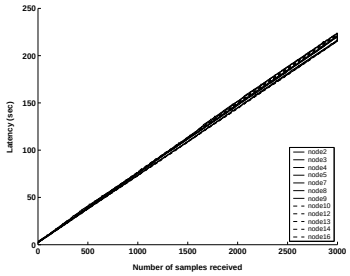

Figure 10: Latencies measured in the seismic-test structure deployment

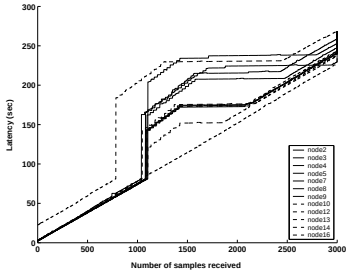

Figure 11: Latencies measured when one StarGate was unplugged during the experiment

NetSHM can be used to rapidly prototype accurate damage detection algorithms. This is important — a lot of structural engineering research uses actual experiments on test structures and a platform like NetSHM has the potential to spur the research conducted by that community.

Damage Localization Results. Columns 3-6 in Table 1 depict the calculated *average* loss of stiffness for each of the four cases over several tests. Consider the result for Case 1 in Table 1, where two braces were removed from the fourth floor. The loss of stiffness on the fourth floor is about 4% after removal of the braces. By contrast, for the other floors the change is less than 1.2%. Thus, there is a clear indication of damage on the fourth floor. The estimated loss of stiffness is also consistent with the level of damage. For example, in Case 2, where all springs on the first floor are removed, the loss of stiffness is almost twice that in Case 1. This observation holds at each floor. Finally, the localization algorithm accurately estimates loss of stiffness in (and only in) the correct floors in Cases 9 and 10.

As seen from Table 1 *every* damage instance was accurately localized as reflected by the significant loss of stiffness (about 5% to 10%) in the corresponding floors and a small variation in the rest of the floors (about 1%). Thus, it is relatively easy, on this structure, to pick out a threshold that will result in no false positives or false negatives in damage localization. As an aside, there are several entries in Table 1 with a negative loss of stiffness. We attribute this to sensor error, as well as numerical errors in the computation. The fact that we are able to actually localize damage on the appropriate floors even with multiple damaged locations (cases 9 and 10) is highly encouraging both as a validation of NetSHM as well the applicability of dense sensing for this application domain.

Finally, we try to quantify the communication savings that could have been obtained if local computation were used in our NetSHM prototype. For damage detection, we find that local computation reduces messaging cost by a factor of 500. In recent work [18], we show how this localization application can be modified to locally compute auto-correlation coefficients at a lower tier node; with this approach, messaging cost is reduced by a factor of 250. We computed these numbers by using the raw sensor readings received during our two experiments, then computing the number of packets necessary to transmit the modal frequencies (for damage detection), or the auto-correlation coefficients (for localization).

3.4 NetSHM at Scale

Finally, we deployed a 14-mote, 2-stargate NetSHM prototype on the seismic test structure shown in Figure 4. The NetSHM application was implemented on a laptop. Figure 9 shows the layout of our network and also depicts the topology for the duration of the experiment. In the figure, dark solid arrows depict dominant links (links over which more than 40% of the packets were transmitted) to a Stargate, while dark dashed arrows depict the less frequently used links. Motes 9 and 13 used both the Stargates to send a significant fraction of the packets at different times during the experiment, while the other nodes were connected to only one of the Stargates for most of the time. The gray dashed arrows depict the links used in the lower-tier *i.e.,* multi-hop routes among the Mica-Z motes. The bi-directional arrows indicate that the link shifted its direction during the experiment. The laptops and the Stargates communicated to each other using their 802.11 radios in ad-hoc mode.

We cannot currently induce damage on this large structure, so we tested a NetSHM application that collects the impulse response of the entire structure. The application tasked the motes to collect data for 1 minute at 50Hz (3000 samples) while the structure was manually actuated to generate impulsive excitation. We time-stamped every sample as it was received at the application and measured latency as the time elapsed since the sensors completed the data. Figure 10 depicts the latencies incurred for transporting samples from each of the motes reliably to the PC. As one would expect, the latency increases linearly for successive samples, since all motes were rate-limited to 2 pkts/sec (each packet carried 8 samples). We also conducted a test to demonstrate the robustness of the system by failing one of the Stargates. The latency of route recovery is depicted in Figure 11. Most motes incurred an increase in latency of about 30-100 seconds, some of which can be attributed to the route update interval, and some to packet loss recovery times. Eventually, all samples were recovered correctly at the base station.

4. CONCLUSIONS AND FUTURE WORK

This paper is an attempt to move away from a vision of sensor networks as being necessarily application-specific. Our design and validation of NetSHM suggests that, at least for a class of high-data rate applications, there exists an architecture that presents a fairly generic programming abstraction and where many systems components (routing, reliable transmission *etc.*) can be reused across applications. We conjecture that the NetSHM architecture might

Table 1: Results of the Damage Detection and Localization Algorithms

Damaged test case	Modal frequencies found in Hz	Average estimated %age loss of stiffness			
		floor 1 (k_1)	floor 2 (k_2)	floor 3 (k_3)	floor 4 (k_4)
C0 : all springs intact	2.6, 7.8, 12.0, 14.9	0.0	0.0	0.0	0.0
C1 : 2 springs removed from floor 4	2.6, **7.4**, **7.7**, **11.9**, **14.8**	0.1	0.073	-0.251	**4.371**
C2 : all springs removed from floor 4	2.6, **7.7**, **11.8**, **14.8**	0.24	0.13	-0.18	**8.814**
C3 : 2 springs removed from floor 3	2.6, 7.8, 12.0, **14.7**	-0.23	-0.05	**4.486**	-0.15
C4 : all springs removed from floor 3	2.6, **7.4**, **7.7**, 12.0, **14.6**	0.99	-0.22	**8.74**	-0.31
C5 : 2 springs removed from floor 2	2.6, 7.8, **11.9**, **14.8**	0.68	**4.734**	-0.1	0.3
C6 : all springs removed from floor 2	2.6, 7.8, **11.7**, **14.7**	0.71	**9.54**	-0.28	-0.37
C7 : 2 springs removed from floor 1	2.6, **7.7**, **11.9**, **14.8**	**5.85**	0.04	-0.56	-0.14
C8 : all springs removed from floor 1	2.6, 7.4, 11.9, **14.8**	**10.39**	-0.05	-0.67	-0.13
C9 : all springs removed from floors 3 and 4	2.6, **7.7**, **11.7**, **14.5**	1.13	-0.28	**11.51**	**6.87**
C10 : all springs removed from floors 1 and 4	2.6, **7.0**, **7.4**, **7.7**, **11.6**	**14.7**	-0.46	-0.78	**9.21**

be more broadly applicable to sensor networks in general, but have deferred an examination of this question to future work.

5. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0325875. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

6. REFERENCES

[1] F.-K. Chang, Ed., *Structural Health Monitoring: The Demands and Challenges emph, Proceedings of the Third International Workshop on Structural Health Moni toring, Stanford University, 12-14 September 2001.* CRC Press.

[2] J. P. Lynch, A. Sundararajan, K. H. Law, A. S. Kiremidjian, and E. Carryer, "Power-Efficient Data Management for a Wireless Structural Monitoring System," in *Proceedings of the 4th International Workshop on Structural Health Monitoring*, vol. 1, Stanford, CA, September 15-17 2003.

[3] J. P. Lynch, A. Sundararanjan, K. H. Law, A. S. Kiremidjian, E. Carryer, H. Sohn, and C. H. Farrar, "Field Validation of a Wireless Structural Monitoring System on the Alamosa Canyon Bridge," in *SPIE's 10th Anuual International Symposium on Smart Structures and Materials*, San Diego, CA, USA, March 2003.

[4] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004.

[5] K. Mechitov, W. Y. Kim, G. Agha, and T. Nagayama, "High-Frequency Distributed Sensing for Structure Monitoring," in *Proc. First Intl. Workshop on Networked Sensing Systems (INSS 04)*, 2004.

[6] S. W. Doebling, C. R. Farrar, M. B. Prime, and D. W. Shevitz, "Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in their Vibration Characteristics: A Literature Review," Los Alamos National Laboratory, Tech. Rep., May 1996.

[7] H. Sohn and C. R. Farrar, "Damage Diagnosis Using Time Series Analysis of Vibration Signals," *Smart Material and Structures*, vol. 10, no. 3, pp. 446–451, 2001.

[8] P. Cawley and R. D. Adams, "The locations of Defects in Structures from Measurements of Natural Frequencies," *Smart Material and Structures*, vol. 10, no. 3, pp. 446–451, 2001.

[9] O. S. Salawu and C. Whilliams, "Damage Location using Vibration Mode shapes," in *Proc. of the 12th International Modal Analysis Conference*, 1994, pp. 933–939.

[10] F. E. Udwadia, "Some Uniqueness Results Related to Soil and Building Structural Identification," *SIAM Journal of Applied Mathematics*, vol. 45, pp. 674–685, 1985.

[11] E. G. Straser and A. S. Kiremidjian, "A Modular, Wireless Damage Monitoring System," John A. Blume Earthquake Engineering Center, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA, Tech. Rep., 1998.

[12] K. Chintalapudi, J. Paek, N. Kothari, S. Rangwala, R. Govindan, and E. Johnson, "Embedded Sensing of Structures: A Reality Check," in *The 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA2005)*, Hong Kong, August 2005.

[13] V. Kottapalli, A. Kiremidjian, J. P. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei, "A Two-Tier Wireless Sensor Network Architecture for Structural Health Monitoring," in *Proc. of SPIE's 10th Annual Symposium on Smart Structures and Materials*, San Diego, CA, March 2003.

[14] M. Maroti, B. Kusy, and G. S. A. Ledeczi, "The Flooding Time Synchronization Protocol," in *Proceedings of the 2nd ACM Sensys*, 2004.

[15] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Issues for Reliable Multihop Routing in Wireless Sensor Networks," Los Angeles, CA, November 2003.

[16] K. Chintalapudi, E. A. Johnson, and R. Govindan, "Structural Damage Detection using Wireless Sensor-Actuator Networks," in *Proc. of 13th Mediterranean Conference on Control and Automation*, Cyprus, June 2005, pp. 27–29.

[17] J. M. Caicedo, S. J. Dyke, and E. A. Johnson, "Natural Excitation Technique and Eigen System Realization Algorithm for Phase I of the IASC-ASCE Benchmark Problem: Simulated Data," *Journal of Engineering Mechanics*, January 2004.

[18] K. Chintalapudi, T. Fu, R. Govindan, and E. Johnson, "Structural Damage Detection and Localization using Wireless Sensor Networks with Low Power Consumption," in *Proc. of the 5h International Conference on Structural Health Monitoring*, Palo Alto,CA, 2005.