ENABLING FRAMEWORK FOR STRUCTURAL HEALTH MONITORING USING SMART SENSORS

J.A. Rice,[1,*,†] K.A. Mechitov,[2] S.H. Sim,[1] B.F. Spencer, Jr.[1] and G.A. Agha[2]

[1] *Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, USA*

[2] *Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA*

ABSTRACT:

Structural Health Monitoring (SHM) is an important tool for the ongoing maintenance of aging infrastructure. The ultimate goals of implementing an SHM system are to improve infrastructure maintenance, increase public safety, and minimize the economic impact of an extreme loading event by streamlining repair and retrofit measures. Networks of wireless smart sensors offer tremendous promise for accurate and continuous structural monitoring using a dense array of inexpensive sensors; however, hurdles still remain. While smart sensors have been commercially available for nearly a decade, full-scale implementation for civil infrastructure has been lacking with the exception of a few short-term demonstration projects. This slow progress is due in part to the fact that programming smart sensors is extremely complex, putting the use of these devices for all but the simplest tasks out of the reach of most engineers. This paper presents an enabling, open-source framework for structural health monitoring using networks of wireless smart sensors. The framework is based on a service-oriented architecture that is modular, reusable, and extensible, thus allowing engineers to more readily realize the potential of smart sensing technology. To demonstrate the efficacy of the proposed framework, an example SHM application is provided.

KEYWORDS: Structural health monitoring, smart sensors, wireless sensors, service-oriented architecture.

---

[*] Corresponding Author
[†] Email: jarice@illinois.edu

1. INTRODUCTION

Structural health monitoring (SHM) provides the means for capturing structural response and assessing structural condition for a variety of purposes. For example, the information from an SHM system can be used to fine-tune idealized structural models, thereby allowing more accurate prediction of the response due to extreme loading conditions, such as an earthquake [1]. SHM also can be used to characterize loads in situ, which can allow the detection of unusual loading conditions as well as validate the structure's design. In addition, real-time monitoring systems can measure the response of a structure before, during and after a natural or man-made disaster, and may be used in damage detection algorithms to assess the post-event condition of a structure.

Given the size and complexity of many civil structures, a large network of sensors is usually required to adequately assess the structural condition. Traditional structural monitoring systems have been moving in the direction of dense deployment in recent years; however, the cost of installation can be thousands of dollars per sensor channel [2], and the amount of data generated by such a system can render the problem intractable [3]. Networks of wireless smart sensors have the potential to improve SHM dramatically by allowing for dense networks of sensors employing distributed computing to be installed on a structure [4,5]. As defined herein, a smart sensor is a battery-powered sensing node with a micro-processor, memory, and a radio transmitter.

While smart sensor technology has been commercially available for nearly a decade, full-scale implementation has been lacking with the exception of a few short-term demonstration projects [7,8]. This slow progress is due in part to the fact that programming smart sensors is extremely complex, putting the use of these devices for all but the simplest tasks out of the reach of most engineers. Moreover, critical issues inherent in smart sensor networks (SSNs), such as synchronized sensing and data loss [9], must be addressed. In addition, the numerical algorithms required for system identification and damage detection must be implemented on sensor nodes which have limited resources. The result is that SHM applications require complex programming, ranging from network functionality to algorithm implementation. Applications software development is made even more difficult by the fact that many smart sensor platforms employ special-purpose operating systems without support for common programming environments. The extensive expertise required to develop SHM applications has severely limited the use of smart sensing technology.

This research tackles the complexity associated with creating SSN applications by developing an open-source framework for structural health monitoring using the design principles of *service-oriented architecture* (SOA) which are described herein. This framework provides a suite of services implementing key middleware infrastructure necessary to provide high-quality sensor data and to transport it reliably across the sensor network, as well as a broad array of SHM algorithms (see http://shm.cs.uiuc.edu/software.html). As these services are loosely coupled and dynamically composable, different SHM applications can be easily created and extended. Because it can be augmented with services for other domains, the framework also provides a common, extensible platform for SSN application development. By leveraging this framework, engineers may focus their attention on the advancement of SHM approaches and the development SHM systems without having to concern themselves with low-level networking, communication and numerical sub-routines. To demonstrate the efficacy of the proposed framework, an example SHM application is provided.

2. SERVICE-ORIENTED ARCHITECTURE

With the exponential growth in available computing power over the last 50 years, the complexity of computer software has likewise increased dramatically. Advances in the fields of programming language design and software engineering allow application developers to deal with this complexity by dividing the software system into smaller, manageable parts. Notably, *object-oriented programming*, which encapsulates data together with the methods used to operate on it, and *component-based software architecture*, which proposes building applications as a composition of self-contained computing components, have been instrumental to the design and development of large-scale software systems. Expanding on these ideas, *service-oriented architecture* has recently been proposed as a way to bring this design philosophy to building dynamic, heterogeneous distributed applications spanning the Internet [10,11]. The following paragraph outlines the core design principles of SOA systems.

SOA design principles are focused on how services are defined and the manner in which data is passed from service to service. Services, in SOA terminology, are self-describing software components in an open or modifiable distributed system. The description of a service, called a *contract*, lists its inputs and outputs, explains the provided functionality, and describes non-functional aspects of execution (timeliness, resource consumption, cost, *etc.*). Data is passed among the services in a common format. An application built using SOA consists of a composition of a number of linked services within a middleware runtime system that provides communication and coordination among them. Unlike traditional component-based architectures, services do not have to be tightly coupled with each other or operate on the same computer; indeed, services do not have to be explicitly linked to each other until execution time. Services do not need to know who provides the required input data, or from where it comes. Different applications can be built from the same set of services depending on how they are linked and on the execution context [12]. This approach provides support for dynamic, highly adaptive applications without the need to revisit and adapt the implementation of each service in a particular application context.

SOA design principles may be applied in the sensor network context as well as on the Internet. Smart sensor networks consist of numerous independent nodes, each an embedded computing platform with a processor, memory, and a radio transmitter. As such, SSN applications are by definition distributed and thus require communication and coordination for parts of the application running on different nodes. SOA has been proposed to address the inherent problems in designing complex and dynamic SSN applications [13,14]. Building an application from a set of well-defined services moves much of the complexity associated with embedded distributed computing to the underlying middleware. This approach also fosters reuse and adaptability, as services for a given application domain can be employed by a multitude of applications.

Perhaps more importantly, SOA provides for a separation of concerns in application development. That is, application designers can focus on the high-level logic of their application, service programmers can concentrate on the implementation of the services in their application domain, and systems programmers can provide middleware services (*e.g.*, reliable communication, time synchronization, data aggregation, *etc.*) that enable the services to interact. In sensor networks, which at this stage are principally used by scientists and engineers, the application designer is likely to be the user of the application as well. This situation makes it especially important for the high-level design of the application and the domain-specific algorithms used by the services to be separated from the low-level infrastructure necessary to make the system work. SOA in SSNs makes it possible to compose and deploy, on-the-fly, complex applications through a web-based user interface suitable for non-programmers [15]. User-driven SSN programming is a relatively young research area with few working implementations, but it holds the promise to lower the barriers to entry in sensor network application development and to accelerate their use in structural health monitoring applications.

3. SOA FOR SHM APPLICATIONS

The proposed service-based framework (http://shm.cs.uiuc.edu/software.html) provides an open-source software library of customizable services for, and examples of, SHM applications utilizing SSNs. SHM middleware services and distributed damage detection algorithms reported in Nagayama *et al.* [4] and Nagayama and Spencer [6], along with a rich array of tools, utilities, and algorithms, have been implemented to enable efficient development of robust, extensible, and flexible structural health monitoring applications on wireless smart sensor networks.

3.1 *Wireless Sensor Platform*

The wireless sensor platform used in this research is the Imote2 (see Fig. 1), which is the only commercially available smart sensor platform that can meet the demands of SHM applications. It has a low-power X-scale processor (PXA27x) with variable processing speed to optimize power consumption. It incorporates a ChipCon 2420 802.15.4 radio with an onboard antenna (Antenova Mica SMD). The onboard memory of the Imote2 is one of the features that sets it apart from other smart sensor platforms and allows its use for the high-frequency sampling required for dynamic structural monitoring. It has 256 KB of integrated RAM, 32 MB of external SDRAM, and 32 MB of flash memory [16].

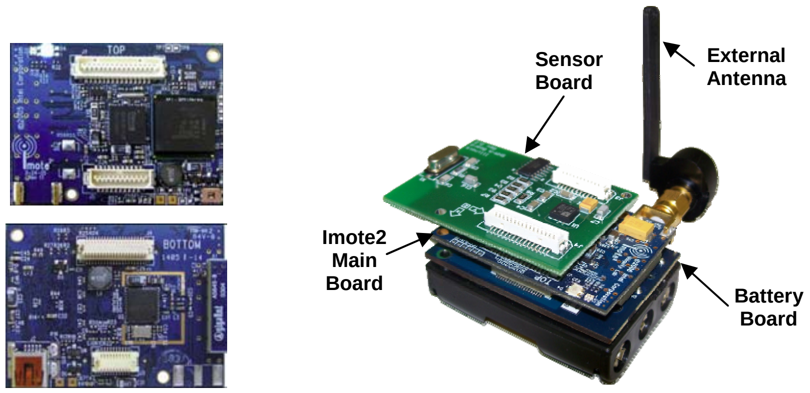

Figure 1. Top and bottom of Imote2 main board (left) and stackable configuration (right).

As with many wireless sensor platforms, the Imote2 employs TinyOS as its operating system. TinyOS is tailored to the specific constraints of sensor networks. In particular, it occupies a small memory footprint while efficiently supporting complex programs. TinyOS applications are implemented in NesC, a C-like programming language which supports the concurrency model of TinyOS [20]. While TinyOS has been adopted by many sensor network applications and has quite a large user-community, developing application code can be a daunting undertaking for engineers lacking such specific programming experience.

3.2 SHM Application Background

The middleware services developed by Nagayama and Spencer [6] and Nagayama *et al.* [4] provide a critical foundation for building a successful SSN application, regardless of whether the goal is to detect damage, determine structural modal properties, or implement some other monitoring strategy. For example, the Distributed Computing Strategy (DCS) [5], which seeks to detect structural damage with a decentralized approach, illustrates how an application utilizes the enabling middleware services provided by the proposed framework. The details of the DCS implementation on the Imote2 can be found in Nagayama and Spencer [6]; for illustration purposes, it is briefly summarized here.

First, the sensor network is divided into overlapping clusters of sensors (Fig. 2), which monitor local portions of the structure. One of the nodes in each cluster, or local sensor community, is assigned as the cluster head and coordinates communication and data processing within the community. The cluster head also communicates with the cluster heads of neighboring communities. When signaled to do so, all nodes in the network simultaneously measure acceleration and synchronize the data.

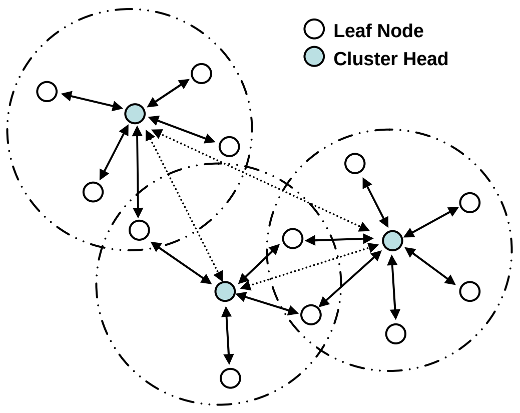

Figure 2. Network topology.

Upon the completion of synchronized sensing, the following tasks occur within each local sensor community [6]:

1. Acceleration data is shared to perform model-based data aggregation using the Natural Excitation Technique (NExT).
2. System identification is performed to determine the dynamic characteristics of the subcomponent of the structure using Eigensystem Realization Algorithm (ERA).
3. The Stochastic Damage Locating Vector (SDLV) damage detection algorithm is applied to determine if the structure has sustained any damage and to determine the location of the damage.

Finally, the cluster head of each local sensor community compares its results with the cluster heads of adjacent, and overlapping, communities to ensure that the results are consistent. If consensus between the cluster heads is achieved, only the outcome of the damage detection method is forwarded to the base station. This entire process is illustrated in the simplified flowchart shown in Fig. 3.

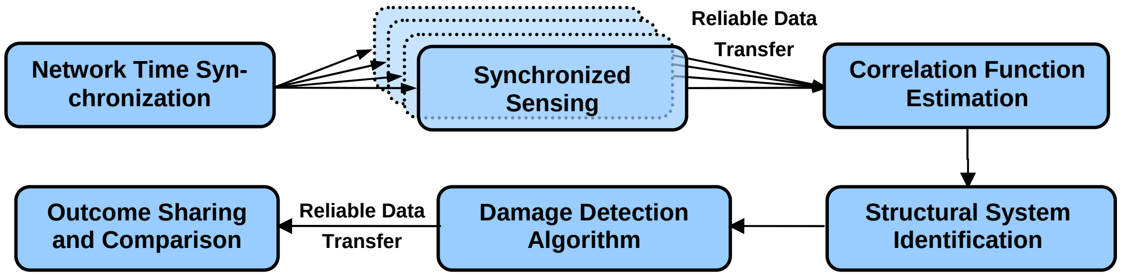
Network Time Synchronization
Synchronized Sensing
Reliable Data Transfer
Correlation Function Estimation
Structural System Identification
Damage Detection Algorithm
Reliable Data Transfer
Outcome Sharing and Comparison

Figure 3. Flowchart of DCS implemented on a network of Imote2s.

This example demonstrates the complexity of typical distributed SHM applications and how the middleware and numerical services can be utilized in their construction. The following section describes the SOA that has been created to provide the necessary modularity and flexibility to facilitate the building of SHM applications.

3.3 Implementation of SOA for SHM

The components of the proposed service-based framework can be divided into three primary categories: (1) foundation services, (2) application services, and (3) tools and utilities. In addition, a library of supporting numerical functions that are common to many SHM algorithms is provided; including fast Fourier transform (FFT), singular value decomposition, Eigenvalue analysis, etc.

3.3.1 Foundation Services

In SOA terminology, services are high level, self-describing building blocks for distributed computing applications. The foundation services implement functionality needed to support the application and other services, rather than implementing application-specific tasks. They include gathering synchronized sensor data, reliably communicating both commands and long data records, and providing accurate and precise timestamps to collected data. In combination, these services can be used by applications to achieve synchronized sensing from a network of sensors.

The *Unified Sensing* service provides a convenient, general-purpose application programming interface (API), replacing the standard TinyOS sensing interface for the Imote2 and extending its functionality to include precise timestamping of the data and providing transparent support for a variety of sensor boards. Data for all sensor channels, together with a single set of associated timestamps, is returned to the application in a single, shared data structure. A compact data representation format is used, which encapsulates all information necessary to recreate the sensor values, yet is memory-efficient for storage and transportation across the wireless network. This complete and self-contained data representation makes it easy to pass around and modify the data without hard-coding connections between components that use only parts of this data [24]. This approach facilitates data being passed directly to the application services described below.

A *Time Synchronization* service provides consistent, network-wide global timestamps for sensor data, making it possible to meaningfully compare data collected from multiple sensors. The *Resampling* application service, described in the next section, then takes the globally-timestamped sensor data and resamples it to a specified fixed sampling rate. This resampling is accomplished in a memory-efficient way with a resampling filter that is applied to the data one block of at a time [6], so that additional memory requirements for the service are independent of the size of the input data.

Because sensor data loss is intrinsic to wireless systems and undermines the ability to perform system identification and detect damage [9], a *Reliable Communication* service has been developed that eliminates data loss when sending commands and data between sensor nodes. The *ReliableComm* service employs four distinct reliable communication protocols, chosen automatically based on the type of communication, to eliminate data loss in an efficient manner.

3.3.2 Application Services

These services provide the numerical algorithms necessary to implement SHM applications on the Imote2s and may also be used independently. For each application service, an application module to test the algorithm on both the PC and the Imote2 has been developed. The numerical services are as follows:

- *SyncSensing*: Resamples timestamped sensor data from a node in a synchronized sensor network (provided by the *Unified Sensing* application service) so that the output for each node in the network has a common sampling rate with a common start time [6]. The service takes raw sensor data and a sparse set of associated timestamps as arguments and applies the resampling filter to the data one block at a time.
- *CFE*: Returns the Correlation Function Estimate (CFE) via FFT calculation. CFE uses two synchronized discrete-time signal vectors to obtain their CFE with user-specified number of FFT points, number of averages, spectral window, and win-

dow overlap. In the NExT approach, the output of CFE can be used as the input to the ERA or SSI system identification services [18].

- *ERA*: Performs the Eigensystem Realization Algorithm (ERA). This time-domain system identification service uses the impulse-response function, or in the case of the NExT algorithm, the correlation functions, to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, modal participation factors, EMAC values, and the state-space matrices defining the identified model of the structure) [19].
- *SSI*: Performs the covariance-driven Stochastic Subspace Identification (SSI) algorithm [25]. This time-domain system identification method uses the cross correlation functions to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, and the state-space matrices defining the model of the structure). Depending on the weighting function, the SSI is classified as (a) Balanced Realization (BR): no weighting, and (b) Canonical Variate Analysis (CVA): natural modes are balanced in terms of energy.
- *SDLV*: Performs output-only, model-based damage detection using the Stochastic Damage Locating Vector (SDLV) method [17]. The inputs of SDLV are the modal characteristics determined by one of the system identification service.
- *FDD*: Performs the Frequency Domain Decomposition (FDD) algorithm [23]. This frequency-domain system identification method uses the cross spectra to determine the modal characteristics of the structure (damped natural frequencies and mode shapes). Because the natural frequencies are selected by a peak-picking method, some modes may not be reliably found.

3.3.3 Tools and Utilities

This section describes application tools and utilities for basic testing and debugging. These are necessary in any large scale or long-term SSN deployments to evaluate network the conditions at the structure, determine appropriate values of adjustable system parameters, and assess power consumption and longevity issues. Included are utilities for resetting nodes remotely, listing the nodes within communication range of the local node, and changing the radio channel and power for local and remote nodes.

The application tools can be categorized as either those operating on a single node or those operate on multiple nodes distributed in the network. The single node application tools include:

- *LocalSensing:* This tool allows sensor data to be collected while a single Imote2 is connected directly to the PC (*i.e.*, no radio communication is required). It allows developers to test the functionality of sensor boards and develop driver software for new boards.
- *imote2comm*: A basic terminal program for interfacing with the Imote2 through the Imote2 Interface Board’s USB port. It uses the serial port UART interface to open a telnet-like connection with the mote.
- *TestServices*: An example program that combines application services: CFE, ERA, and SDLV. It uses acceleration signals as input in the CFE service to calculate the correlation functions that is used in the ERA service. The estimated modal characteristics of the structure are then used in the SDLV service to identify damage.

The application tools that involve multiple nodes are as follows:

- *TestRadio*: Tests the raw bidirectional communication between a sender node and a group of receiving nodes, and output the packet loss rate (in each direction, and round-trip).
- *RemoteSensing:* A network-wide distributed application, this tool is used to collect sensor data from multiple sensors. The network is synchronized prior to sensing, then timestamped data is collected. Depending on the command that is given at run time, this service can output either the raw timestamped data or resampled synchronized data. If the resampling option is selected, the data is resampled locally using the *SyncSensing* service to account for any jitter or non-

uniform delay in the start of sensing for each node. All data and commands in *RemoteSensing* are sent between nodes using the *ReliableComm* service, eliminating data loss.

- *DecentralizedDataAggregation*: This sample application illustrates use of the proposed framework data acquisition and processing based on decentralized hierarchical sensor network. This application supports multiple sensor clusters, in which data processing is conducted independently to other clusters. The main outputs of the application are sensor data and their correlation functions in each sensor cluster.

The *RemoteSensing* and *DecentralizedDataAggregation* application tools employ a distributed state machine to determine the timing and control flow of the application across a network of sensors. A state machine is a formal method for defining how an application behaves or responds when it is in a particular *state* and the *transitions* required to move between states. The flowchart given in Fig. 4 illustrates the state machine for the *RemoteSensing* application. Table 1 summarizes each state and transition associated with *RemoteSensing*.

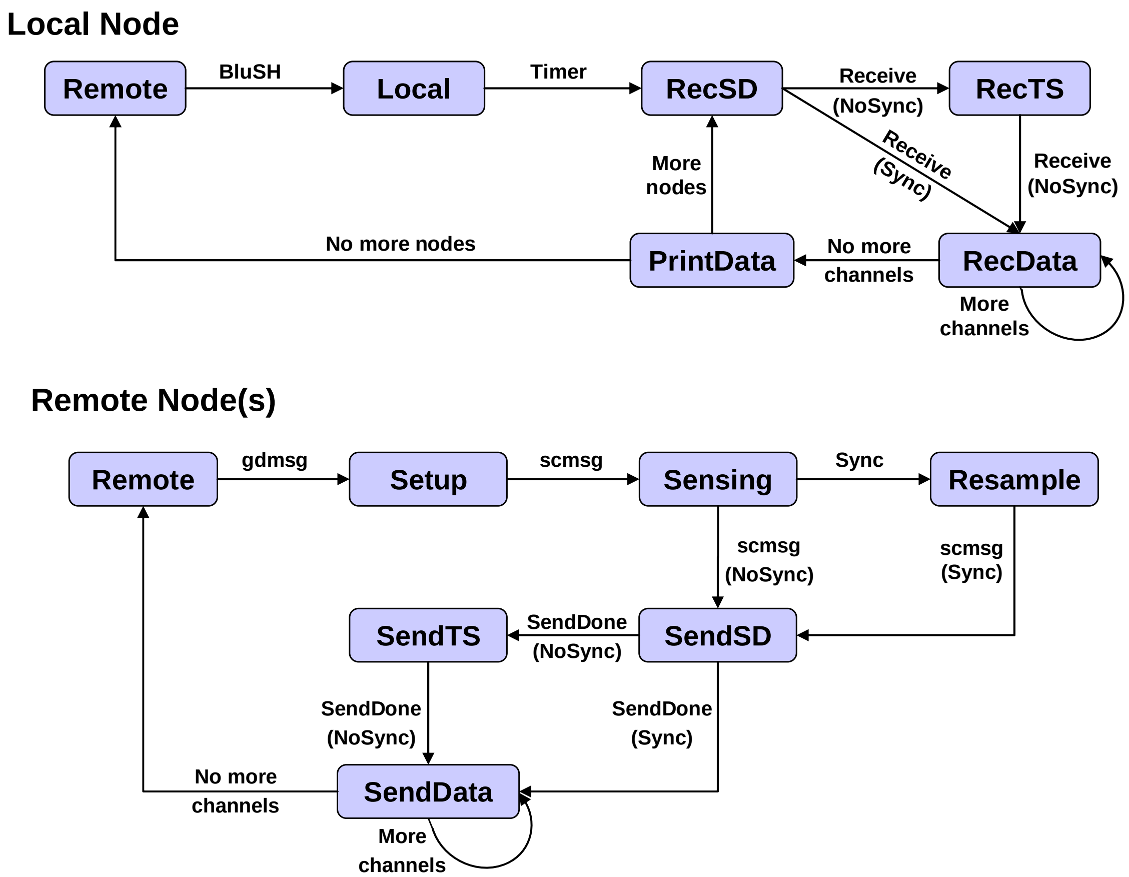

Figure 4. *RemoteSensing* State machine for the local node (top) and remote nodes (bottom). Boxes represent states, arrows represent transitions, and arrow labels indicate conditions or actions needed for the transition to occur.

The components of the service-oriented architecture described above lend themselves to the exploration of new approaches to solve specific problems. As a simple example, the figure below represents how the system identification method can be swapped out in an SHM application. In keeping with the SOA framework, these interchangeable services share the same input and output parameters. Other application examples that can benefit from the modular services provided in the framework include distributed damage detection algorithms that rely only on the parameters derived from the correlation function estimates [22] or methods for distributed modal parameter estimation in a SSN [23].

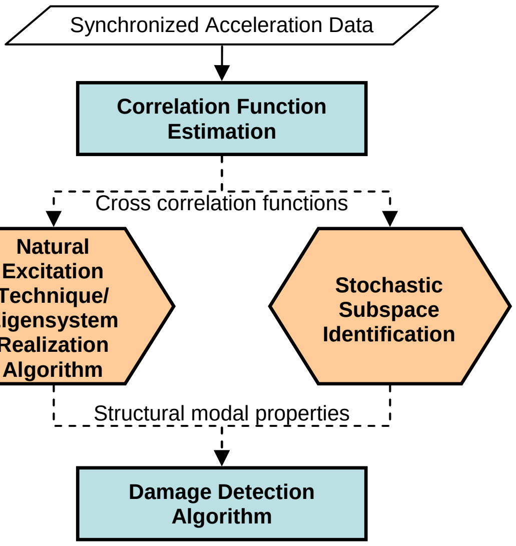
Synchronized Acceleration Data
Correlation Function Estimation
Cross correlation functions
Natural Excitation Technique/ Eigensystem Realization Algorithm
Stochastic Subspace Identification
Structural modal properties
Damage Detection Algorithm

Figure 5. Alternate services for SHM application development.

4. EXAMPLE APPLICATION

To demonstrate the efficacy of the proposed service-based framework, this section describes development of *DecentralizedDataAggregation*, a decentralized application for measurement, aggregation, and compression of sensor data using the proposed services. *DecentralizedDataAggregation* provides a clear example of how the proposed framework facilitates the assembly of the services into a full application by implementing the necessary control logic. In particular, this example illustrates the coordination of a distributed application by showing how an application's input parameters are specified by the user (at compilation and/or at run time), how each mote is functionally differentiated, how data is passed between services, and how the scheduling of tasks take place.

The design of an application for SSNs (*e.g.*, *DecentralizedDataAggregation*), requires careful consideration of the network topology. Because a centralized data acquisition system is not feasible for densely deployed smart sensors, a decentralized processing scheme is preferred, in which data communication and processing occur in each sensor cluster independently [6]. Because each modal analysis method requires different inputs (*i.e.*, correlation function or cross spectrum with respect to single or multiple references), the network topology should be tailored to the modal analysis method considered in SSN applications. *DecentralizedDataAggregation* is designed to calculate the correlation function with a single reference that can be used in ERA.

The *DecentralizedDataAggregation* application consists of four main parts: (a) initialization (b) synchronized sensing, (c) data processing (estimation of correlation function), and (d) collection of the processed data to the base station (see the flow chart in Fig. 6). In the initialization stage, the user-defined parameters, mainly specifying the data processing and sensor topology, are disseminated from the base station to the sensor nodes. To realize decentralized data processing, *DecentralizedDataAggregation* allows a sensor topology that consists of multiple local sensor communities with overlapping nodes shared by neighboring sensor communities. For the sensing part, the *Time Synchronization*, *Unified Sensing,* and *SyncSensing* services are combined to measure synchronized data as in *RemoteSensing*. In the data processing part, the *CFE* service utilizes the sensor data to calculate the correlation functions in each sensor community. The correlation functions are first sent to the cluster head of each sensor community. If services such as *ERA* and *SDLV* are implemented, the correlation functions collected at the cluster heads can be utilize to estimate modal properties (*e.g.*, natural frequencies, modal damping factors, and mode shapes) and damage information. In every data communication between the nodes including the base station, the *Reliable Communication* service is used. The state machine shown in Fig. 7 and described in Tables 2, 3, and 4 illustrates this process. Integrating the services provided by the SOA framework into the desired network topology, the SHM applications can be efficiently developed.

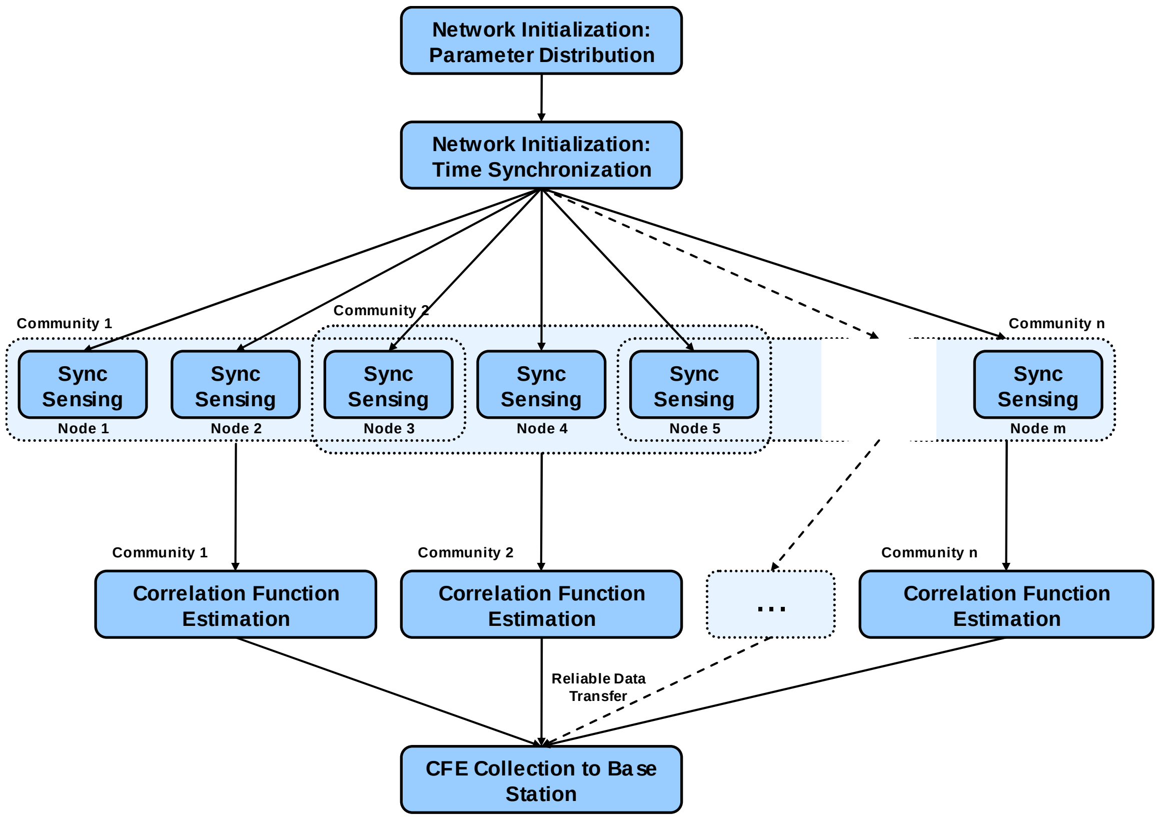
Network Initialization: Parameter Distribution
Network Initialization: Time Synchronization
Community 1
Community 2
Community n
Sync Sensing
Node 1
Node 2
Node 3
Node 4
Node 5
Node m
Community 1
Community 2
Community n
Correlation Function Estimation
. . .
Reliable Data Transfer
CFE Collection to Base Station

Figure 6. Flowchart of *DecentralizedDataAggregation*.

The correlation functions calculated in a decentralized manner on the Imote2s are compared to correlation functions calculated in MATLAB on the same sensor data. Fig. 8 shows that both the auto- and cross-correlation functions produced by *Decentralized-DataAggregation* are identical to those produced on the PC.

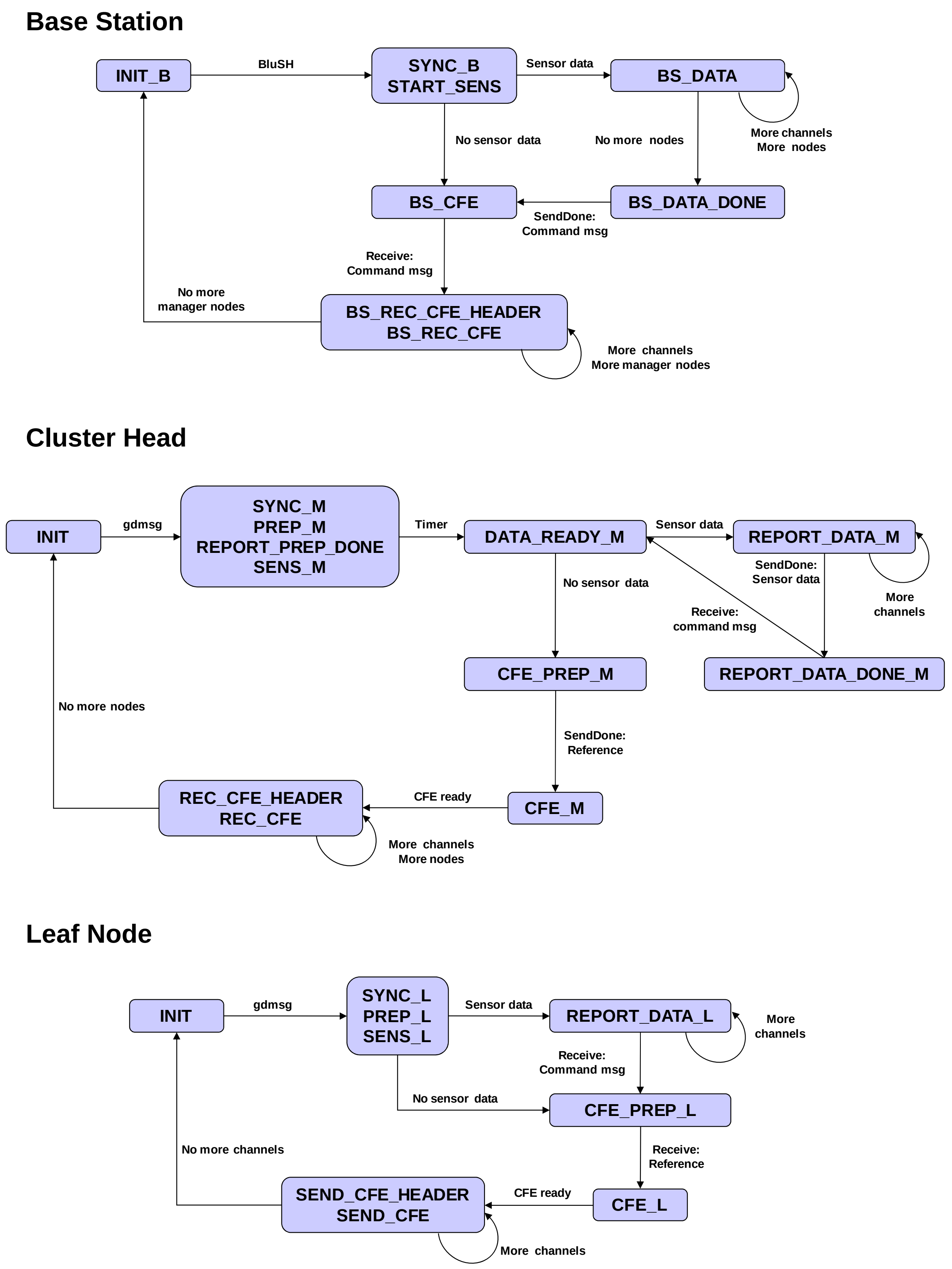

Figure 7. ***DecentralizedDataAggregation*** **State machine for the base station (top), cluster head (middle) and leaf nodes (bottom)**

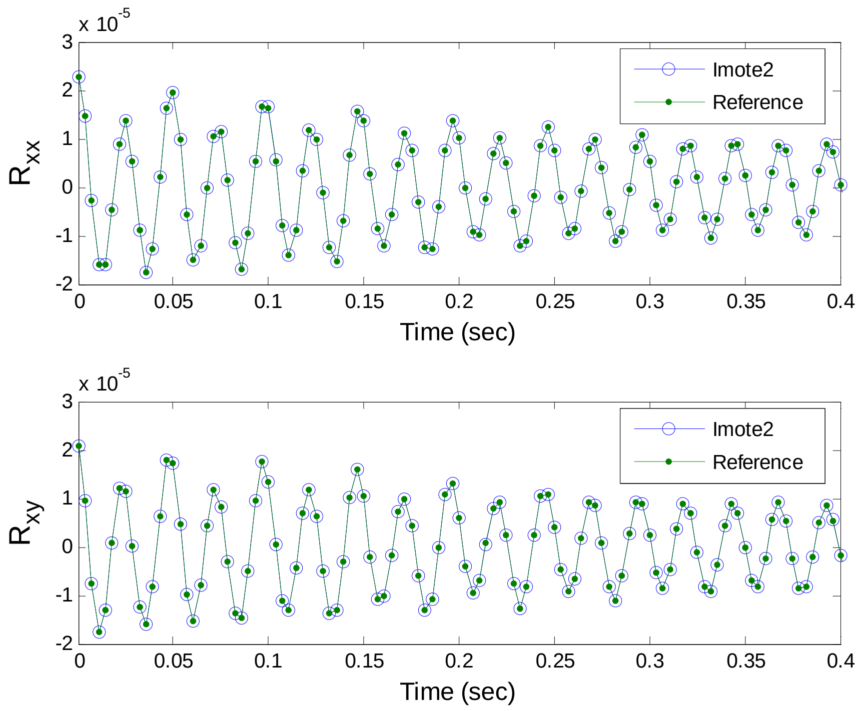

Figure 8. Comparison between auto-correlation (top) and cross-correlation (bottom) functions calculated in MATLAB and on the Imote2 using *DecentralizedDataAggregation.*

Because the output of *DecentralizedDataAggregation* can be used as the input *for other services*, *DecentralizedDataAggregation* provides the foundation for decentralized modal analysis and damage detection in a network of smart sensors. For example, the single-reference correlation functions that it produces may be used directly by the *ERA* service for modal property estimation. For the output of *DecentralizedDataAggregation* to be used with the FDD and SSI/CVA services, its data processing scheme should be extended to facilitate the use multiple reference sensors. Then, the output modal properties produced (i.e., by the ERA, FDD, or SSI/CVA) can be utilized in damage detection service such as SDLV.

5. CONCLUSIONS

This paper described an open-source framework developed using the design principles of service-oriented architecture (SOA). This SOA-based approach creates an enabling framework that manages the complexity inherent in the use of SSNs. As a result, researchers and application engineers can design and deploy efficient SHM systems without worrying about how the underlying middleware and application services are implemented. The paper also provided an example application, *DecentralizedDataAggregation,* to illustrate how the framework components can be assembled to build complex applications. The service-based framework described herein will ensure that smart sensor technology sees more widespread use in SHM applications, ultimately driving the technology forward to improve infrastructure maintenance and enhance public safety.

Both the services provided in the framework and the approach in general can be applied to a broad array of SHM problems; the associated software, documentation, and examples discussed herein are available at http://shm.cs.uiuc.edu/software.html. Work is also underway to automate the process of interconnecting the services and tool to create applications using "drag-and-drop" graphical programming [15].

6. ACKNOWLEDGEMENT

The authors gratefully acknowledge the support of this research by the National Science Foundation, under grant CMS 0600433 (Dr. S. C. Liu, program manager).

REFERENCES

1. Farrar, C.R. and Doebling, S.W. Vibration-based health monitoring and model refinement of civil engineering structures, *Proc. 1st International Architectural Surety Conference* 1997.
2. Celebi, M. Seismic instrumentation of buildings (with emphasis on federal buildings). *Report no.0-7460-68170, United States Geological Survey (USGS)* 2002.
3. Nagayama, T., Spencer Jr., B.F., Rice, J.A. and Agha, G. Smart Sensing Technology: A New Paradigm for Structural Health Monitoring, *Proc., 39th Joint Meeting of the US-Japan Joint Panel on Wind and Seismic Effects, UJNR* 2007.

4. Spencer Jr., B.F., Ruiz-Sandoval, M and Kurata, N. Smart Sensing Technology: Opportunities and Challenges, *Structural Control and Health Monitoring* 2004; **11**, 349–368.

5. Gao, Y. and Spencer Jr., B.F. Structural health monitoring strategies for smart sensor networks, *NSEL Report, Series 011*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/8802, 2008 **.**

6. Nagayama, T. and Spencer Jr., B.F., Structural health monitoring using smart sensors, *NSEL Report, Series 001*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/3521, 2007.

7. Kim S., Pakzad S., Culler D., Demmel J., Fenves G., Glaser S. and Turon, M. Health monitoring of civil infrastructures using wireless sensor networks. *Proc. 6th International Conference on Information Processing in Sensor Networks* 2007*;* 254-263.

8. Lynch, J. P. and Loh, K. A summary review of wireless sensors and sensor networks for structural health monitoring. *Shock and Vibration Digest* 2006; **38(2)**, 91-128.

9. Nagayama, T., Sim, S-H., Miyamori, Y., and Spencer Jr., B.F. Issues in structural health monitoring employing smart sensors, *Smart Structures and Systems* 2007; **3(3)**, 299-320.

10. Singh, M.P. and Huhns, M.N. Service-Oriented Computing: Semantics, Processes, Agents, John Wiley and Sons, New Jersey.

11. Tsai, W.T. (2005). Service-Oriented System Engineering: A New Paradigm. Proc. IEEE International Workshop on Service-Oriented Systems Engineering, 3-8.

12. Gu, T., Pung, H.K. and Zhang, D.Q. A service-oriented middleware for building context-aware services. *J. Network and Computer Applications* 2005; **28(1)**, 1-18.

13. Liu, J. and Zhao, F.. Towards semantic services for sensor-rich information systems. *Proc. International Workshop on Broadband Advanced Sensor Networks*.

14. Mechitov, K., Razavi, R., and Agha, G.. Architecture design principles to support adaptive service orchestration in WSN applications. *ACM SIGBED Review* 2007; **4(3)**.

15. Razavi, R., Mechitov, K., Agha, G. and Perrot, J.-F.. Ambiance: A Mobile Agent Platform for End-User Programmable Ambient Systems. *Advances in Ambient Intelligence, Frontiers in Artificial Intelligence and Applications* 2007; **164**, 81-106.

16. Intel Corporation Research, *Intel Mote2 Overview, Version 3.0,* Santa Clara, CA, 2005.

17. Bernal, D. (2006). Flexibility-Based Damage Localization from Stochastic Realization Results, *J. of Engineering Mechanics* (2006); **132(6)**, 651-658.

18. James, G. H., Carne, T. G., & Lauffer, J. P. The natural excitation technique for modal parameter extraction from operating wind turbine, *Report No. SAND92-1666*, UC-*261*, Sandia National Laboratories, 1993.

19. Juang, J.-N. and Pappa, R. S. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics* 1985; **8(5)**, 620-627.

20. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. TinyOS: An Operating System for Sensor Networks. *Ambient Intelligence*, Weber, W., Rabaey, J.M., Aarts, E., Eds., Springer, Berlin, Heidelberg, 2005; 115-148,.

21. Nagayama, T., Spencer Jr., B.F., Mechitov, K., Agha, G. Middleware services for structural health monitoring using smart sensors, *Smart Structures and Systems*, 2008 (in press).

22. Castaneda, N., Sun, F., Dyke, S., Lu, C., Hope, A., Nagayama, T. Implementation of a Correlation-based Decentralized Damage Detection Method Using Wireless Sensors. *Proc. 2008 ASEM Conference*, Jeju, Korea, 2008.

23. Brincker, R., Zhang, L., and Anderson, P. Modal identification of output-only systems using frequency domain decomposition. *Smart Mater. Struct*., 2001; **10(3)**, 441–445

24. Rice, J.A., Spencer Jr., B.F., Mechitov, K. and Agha, G. Flexible Smart Sensing Framework for Structural Health Monitoring, *Proc. US-Korea Workshop on Smart Structures Technology* 2008.

25. Hermans, L. and Van Der Auweraer, H. (1999) "Modal testing and analysis of structures under operational conditions: Industrial applications." Mechanical Systems and Signal Processing, 13(2), 193-216.

Table 1. State and transitions for *RemoteSensing* application.

State	*Description*
Remote	Initial state
Local	Initial local node state
Setup	Receive and store sensing parameters
Sensing	Data acquisition
Resample	Resample of acquired data based on timestamps and initial delay
SendSD	Send sensor data structure
RecSD	Receive sensor data structure
SendTS	Send timestamps (if data is not resampled)
RecTS	Receive timestamps (if data is not resampled)
SendData	Send sensor data
RecData	Receive sensor data
PrintData	Write data to PC
Transition	
BluSH	Application initialized by user through the Blue Shell interface
gdmsg	*GetData* message containing sensing parameters received
Timer	Timer set to wait for remote node(s) to acquire data
scmsg	*StartCollection* or request for data message received
Sync	Resampling flag set
NoSync	Resampling flag not set
sendDone	Previous message sent successfully
receive	Data successfully received

Table 2. State and transitions for *DecentralizedDataAggregation* application operating on the base station

State	
INIT_B	Initial state
SYNC_B	Time synchronization
START_SENS	Send a command to start sensing
BS_DATA	Receive sensor data
BS_DATA_DONE	Change state according to request
BS_CFE –	Preparation for CFE
BS_REC_CFE_HEADER	Receive CFE header
BS_REC_CFE	Receive CFE
Transition	
BluSH	Application initialized by user through the Blue Shell interface
Sensor data	Sensor data request flag set
SendDone	Previous message or data sent successfully
Receive –	Previous message or data received successfully

Table 3. State and transitions for *DecentralizedDataAggregation* application operating on the cluster heads.

State	
INIT	Initial state
SYNC_M	Time synchronization
PREP_M	Channel preparation for sensing
REPORT_PREP_DONE	Report to the base station that all channels are ready
SENS_M	Sensing
DATA_READY_M	Sensing and resampling for synchronized sensing
REPORT_DATA_M	Send sensor data to the base station
REPORT_DATA_DONE_M	Receive a command to proceed from the base station
CFE_PREP_M	Check all leaf nodes are ready for CFE calculation
CFE_M	Calculate CFE
REC_CFE_HEADER	Receive CFE header from leaf nodes
REC_CFE	Receive CFE from leaf nodes
Transition	
gdmsg	Message containing parameters received
Sensor data	Sensor data request flag set
SendDone	Previous message or data sent successfully
Receive	Previous message or data received successfully
CFE ready	CFE for all sensors in the local group is ready

Table 4. State and transitions for *DecentralizedDataAggregation* application operating on the leaf nodes.

State	
INIT	Initial state.
SYNC_L	Time synchronization.
PREP_L	Channel preparation for sensing.
SENS_L	Sensing.
REPORT_DATA_L	Send sensor data to the base station.
CFE_PREP_L	Communicate with manager nodes.
CFE_L	Calculate CFE.
SEND_CFE_HEADER	Send CFE header from leaf nodes.
SEND_CFE	Send CFE from leaf nodes.
Transition	
gdmsg	Message containing parameters received.
Sensor data	Sensor data request flag set.
Receive	Previous message or data received successfully.
Command msg	Command message from the manager nodes received.
CFE ready	CFE is ready.