Long-term and Short-term Autonomous Structural Health Monitoring Strategies Using Wireless Smart Sensor Networks*

Jian Li, Kirill A. Mechitov, and Billie F. Spencer, Jr.

***Abstract*— The ability of running autonomously with minimum human intervention is critical to an effective structural health monitoring (SHM) system for continuous monitoring of civil infrastructure. Owing to the on-board computation and wireless communication capabilities, wireless smart sensors (WSSs) offer great flexibility in data acquisition, storage, management, and transmission, but pose challenges in the design of autonomous SHM systems. In this paper, two monitoring scenarios are considered and autonomous monitoring strategies are proposed for each case based on wireless smart sensor networks (WSSNs). The first one is an autonomous WSSN running with a central base station, aiming to provide long-term monitoring of civil infrastructure in an autonomous and continuous manner for an extended period of time (a few months to years). The second system is designed to run autonomously without a base station so that it can be deployed in a more efficient manner for campaign-type monitoring of short-term (a few days to weeks) behavior of civil infrastructure. The ability of running without a base station removes the needs for tethered power supply and therefore enables wider application of SHM systems for civil structures located in remote areas where tethered power is often unavailable.**

I. INTRODUCTION

Aging civil infrastructure needs to be properly maintained to ensure its safety and serviceability. Although visual inspection still remains one of the most important maintenance methods for civil infrastructure, it is time consuming, labor intensive, cost inefficient, and prone to error. One tragic example is the collapse of the I-35W highway bridge over Mississippi River in Minnesota in 2007 killing 13 people and injuring 145 even though it passed a visual inspection a year prior to failure. Structural Health Monitoring (SHM) is an effective technology which aims to assess the structural integrity using sensors and signal processing approaches. More importantly, the advancement of wireless sensing technology has brought SHM to a whole new level by significantly reducing the cost and providing more flexible and powerful functionalities, enabling better understanding of structural integrity through dense instrumentation and in-network data aggregation.

One important aspect of SHM is continuous and autonomous operation. The ability of running autonomously and continuously with minimum human intervention is critical to an effective SHM system. Owing to the on-board computation and wireless communications capabilities, wireless smart sensors (WSSs) offer great flexibility in data acquisition, storage, management, and transmission, but pose challenges in the design of autonomous SHM systems due to the constrained power availability and complex functionality of the network compared with the traditional wired SHM systems. A successful design of wireless SHM systems should provide autonomous and smooth operation during the targeted deployment period while supporting a number of concurrently running applications. In addition, the network operation should run in a way that energy consumption is minimized.

Depending on the duration and the purpose of monitoring effort, SHM systems can generally be classified into two categories, i.e. long-term SHM systems and short-term, or campaign-type SHM systems. Long-term SHM systems are generally in operation for a few months or years, aiming for monitoring long-term deterioration, fatigue damage of structures, and so on. On the other hand, campaign-type SHM systems are deployed for a short amount of time ranging from a few hours to a few days or weeks, in order to assess the current integrity of structures such as load-carrying capacity. Campaign SHM systems are widely applied in assessment of retrofitted structures, post-disaster condition assessment, and design optimization of long-term SHM systems before permanent deployment, etc.

In this paper, two designs of autonomous SHM Systems are proposed for long-term and short-term wireless SHM applications, respectively. For long-term SHM, based on the autonomous monitoring strategy proposed by Rice et al. (2010), major improvements are made in the event-triggering mechanism for network-wide activities (*ThresholdSentry*) and the scheduling method to support autonomous network operation (*AutoMonitor*). For short-term SHM, the system is designed to run without a central base station so

*Resrach supported by the National Science Foundation Grants CMS 06-00433, CMMI-0724172, CMS 09-28886, and CPS 10-35773.

Jian Li is with the Department of Civil, Environmental, and Architectural Engineering, The University of Kansas, Lawrence, KS 66045 USA (Tel: 785-864-6850, e-mail: jianli@ku.edu).

Kirill A. Mechitov is with the Department of Computer Science at the University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mechitov@illinois.edu).

Billie F. Spencer, Jr. is with the Department of Civil and Environmental Engineering, University of Illinois at Urbana- Champaign, Urbana, IL 61801 USA (e-mail: bfs@illinois.edu).

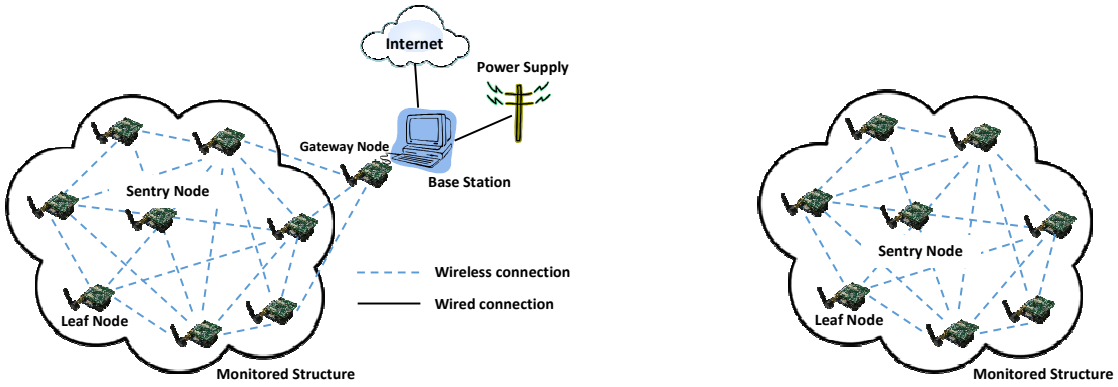
that short-term monitoring can be performed in a more efficient and cost-effective way. More importantly, with this system, SHM can be applied to civil structures located at remote areas where external power supply to support a base station is often unavailable.

II. HARDWARE AND SOFTWARE COMPONENTS

The autonomous SHM systems proposed in this research are based on Imote2 wireless smart sensors. Imote2 sensor is selected because of its low-power X-scale processor (PXA27x) with variable processing speed (13 MHz – 416 MHz), which are essential to facilitating high-frequency sampling and computationally intensive data processing while keeping minimum energy consumption for the overall network operation. Moreover, its relatively large-size onboard memory is another key feature to support data-intensive SHM applications. Imote2 does not have sensing capability. Instead, it utilizes a stackable modular design which allows different types of daughter board to be attached. In this research, the SHM-A sensor board, a multi-metric sensor board (Rice and Spencer, 2009) developed by the Illinois SHM Project (ISHMP), is used to measure 3-axis acceleration, temperature, light, and humidity.

The proposed long-term and short-term continuous SHM systems are illustrated in Fig.1. For the long-term SHM System, a network typically consists of Imote2 leaf nodes monitoring the structure, a gateway node which is also an Imote2 node connected to a central base station. The base station is in turn connected to infrastructure such as external power supply and internet for remote access. The gateway node and base station serve as the main task scheduler and data sink for the network, as well as the interface between the network and network administrators. A subset of leaf nodes can be designated as sentry nodes. As will be discussed later, the leaf nodes will spend most of their time in deep sleep mode in order to conserve energy and wake up periodically for a short amount of time to listen to external commands. The sentry nodes, after waking up, check vibration level and send flags to the gateway node to trigger network-wide activities, such as sensing, cable tension estimation, data aggregation, etc.

On the other hand, for many civil infrastructures, such as bridges and tunnels located at remote areas, setting up a base station can be expensive and time-consuming due to limited access to external power source. Particularly, for campaign-type monitoring in which a sensor network is deployed for a few days or weeks to monitor the current structural condition, it is desirable to install a number of sensor nodes quickly and let them run autonomously without requiring extra effort to set up a base station, as illustrated in Fig. 1(b). The sensor network can collect critical and informative data and store them into flash memory on each sensor node. After the deployment, the sensors can be retrieved and the data can be harvested for subsequent analysis. The strategy can create a much more easy-to-deploy and efficient wireless SHM system for campaign monitoring of civil infrastructure. Therefore, the short-term SHM system proposed in this research runs without a central base station and hence no gateway node is used. In this case, all leaf nodes are treated equally in the network, except that some nodes can be designated as the sentry nodes to trigger network-wide tasks.



(a) long-term SHM (b) short-term SHM

Figure 1. Illustration of (a) long-term autonomous wireless SHM system and (b)short-term base-station-free wireless SHM system

The ISHMP Services Toolsuite (*http://shm.cs.illinois.edu/software.html*) is utilized as the basis for the proposed software development. The toolsuite is a collection of open source middleware and application-level services designed specifically for implementing various SHM applications using Imote2 sensors. The toolsuite is written in NesC (Gay et al., 2003), the same program language that the TinyOS (http://www.tinyos.net) is written in, and is designed based on the concept of Service-oriented architecture (SOA), which divides the software into smaller, more manageable components such that the concerns in application development are separated (Rice and Spencer, 2009; Mechitov, 2011). The toolsuite provides three types of services: (1) foundation services, (2) application services, and (3) tools and utilities. The Foundation services address the basic needs to fulfill a functioning wireless sensing network, such as *time synchronization*, *sensing*, *reliable communication*, *multi-hop routing*, and *SnoozeAlarm* for duty cycling, etc. The application services provide building blocks for SHM applications, such as *Synchronized Sensing*, *Correlation Function Estimation*, *Stochastic Damage Locating Vector (SDLV) method*, *Eigensystem Realization Algorithm (ERA)*, etc. Tools and

Utilities contain the highest-level applications to operate or test the sensor network, such as the *RemoteSensing*, *DecentralizedDataAggregration (DDA)*, *CableTensionEstimation (CTE)*, *AutoUtilsCommand*, and *AutoMonitor*, as well as some test applications. More detailed information about each application can be found in Rice and Spencer (2009), Sim and Spencer (2009), and Mechitov (2011).

III. LONG-TERM AUTONOMOUS MONITORING WITH WSSNS

An initial version of the long-term autonomous monitoring strategy was developed by Rice et al. (2010). Three major components were developed and integrated into the autonomous system, including sleep cycling, threshold triggering, and autonomous monitoring. The corresponding developed software services and tools are *SnoozeAlarm*, *ThresholdSentry*, and *AutoMonitor*, respectively. The system was firstly deployed on the Jindo Bridge in 2009 with 70 Imote2s and SHM-A sensor boards divided into two sub-networks, one with 33 leaf nodes and one with 37 leaf nodes. The second deployment on the Jindo Bridge was made in 2010 (Jo, et al., 2011) with 113 imote2 sensor nodes divided into four sub-networks. In the second deployment, multi-hop communication was adopted in one of the four sub-networks. These two deployments revealed a number of issues which are addressed in this paper. This section describes the main issues in *ThresholdSentry* and *AutoMonitor* and the improved design and implementation are introduced. The details of the Jindo Deployments are also introduced.

2.1. Threshold Triggering

To avoid data inundation and conserve energy, long-term wireless SHM systems do not perform sensing continuously throughout the deployment. Instead, the leaf nodes spend most of their time in deep sleep mode and wake up periodically only for a short amount of time to listen to external commands. These external commands can from either scheduled events or triggered events. The basic idea of threshold triggering is that a subset of the leaf nodes is designated as sentry nodes. These sentry nodes check periodically the vibration level by collecting data for a short duration and calculating the vibration level, which is then compared with a predetermined threshold level. A flag is then sent to the gateway node with the comparison result. If the threshold level is exceeded, the gateway node wakes up the entire network to perform network-wide tasks such as synchronized sensing, cable tension estimation, etc. Otherwise, the sentry nodes go back to the deep sleep mode and repeat the checking of the vibration level when the next threshold checking is due. Threshold triggering allows the network to respond only to the significant events which generate high level of structural vibration. High level structural vibration data offers better information with low noise for damage detection and better understanding of structural characteristics.

In the *Threshold Triggering* service developed by Rice et al. (2010), as shown in Fig. 2, once *ThresholdSentry* is initiated, a check timer is started at the gateway node and is fired after the check interval is reached. The gateway node wakes up the sentry nodes and sends the sensing parameters to the sentry nodes such as sensing duration, sampling frequency, etc. The sentry nodes calculate the vibration levels and send flags to the gateway node. The gateway node then makes decision based on whether the threshold level is exceeded or not. If exceeded, trigger-based network-wide activities are initialized. Otherwise the gateway node starts another check timer and the previous process is repeated.

In this previous design, threshold checking by the sentry nodes is initiated by the gateway node. The gateway node wakes up the sentry nodes to check the vibration level. During the Jindo Bridge deployments, *ThresholdSenry* was working smoothly in single-hop networks in which single-hop communication protocol was utilized. However, in the multi-hop network adopted in the 2010 deployment, *ThresholdSentry* was found to cause power drainage of the leaf nodes. The reason was found later that in the multi-hop routing protocol, to establish routes between the source node and the destination node, the route request (RREQ) message is broadcasted to the network; hence any node which hears the RREQ message will wake up. Details of the multi-hop communication protocol can be found in Nagayama et al. (2010). Therefore, when the gateway node performs a threshold check, often times the entire network wakes up because of the RREQ message. The leaf nodes then stay awake until the watchdog timer puts them back to sleep mode if no network-wide activity is initiated after the threshold check. In the deployments, the *ThresholdSentry* interval was set as 20 minutes which is shorter than the watchdog timer interval (60 minutes). Consequently, the leaf nodes stayed awake all the time during the deployment and had no chance to go back to sleep mode, leading to quick drainage of battery power. Therefore, the initial *ThresholdSentry* design is not able to achieve the goal of conserving energy in multi-hop networks because of the way the routing protocol is designed.

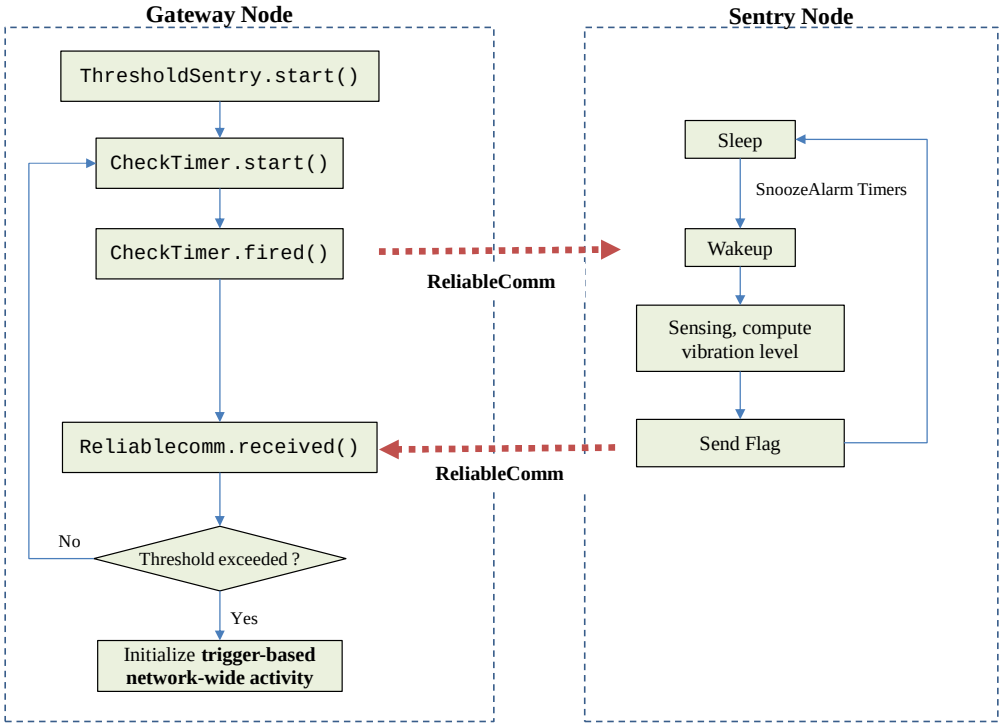


Figure 2. Block diagram of the original ThresholdSentry service

To address the above issue with multi-hop networks, a new design of *ThresholdSentry* is proposed and implemented by switching the control of threshold checking from the gateway node to the sentry nodes. As illustrated in Fig. 3, when *ThresholdSentry* is initiated at the gateway node, the gateway node sends the sentry parameters to the sentry nodes through reliable communication. A metadata block in the flash memory of the sentry nodes is created to store these parameters. The sentry parameters include threshold check interval, sampling frequency, number of data points to be sampled for threshold checking, etc. The sentry nodes then return back to normal sleep/wakeup cycling and check periodically the vibration level autonomously. Since the sentry nodes do not stay awake all the time, scheduling the threshold checking events by the sentry nodes themselves using timers is challenging. To simplify the control logic, the threshold check interval is converted into number of duty cycles (*TgtCount*) based on the given time duration. For example, one duty cycle is composed of 10 seconds of sleep time and 0.5 seconds of wakeup/listen time; therefore, a check interval of 20 minutes is translated into about 114 duty cycles. A parameter stored in the metadata block, *SACount* is used to track the number of duty cycles the sentry node has gone through since the last threshold check and is updated every time before the sentry node returns back to sleep. After waking up from sleep mode, the sentry node compares *SACount* with *TgtCount*. If *SACount* has not yet reached *TgtCount*, it still listens to other external commands and performs relevant tasks if necessary before returning back to sleep mode. If *SACount* is equal to *TgtCount*, the sentry node performs sensing and computes the vibration level. If the threshold level is exceeded, a flag is sent back to the gateway node to initiate network-wide tasks. The metadata is then updated by setting *SACount* back to zero. If the threshold level is not exceeded, the metadata is also updated and the sentry node returns back to the sleep/wakeup cycling and the above process is repeated.

Unlike the previous *ThresholdSentry* design in which the communication between the sentry nodes and the gateway node is centrally coordinated by the gateway node, the sentry nodes in the new design run independently so communication conflict may occur if multiple sentry nodes are sending flags to the gateway node concurrently. To avoid such potential conflict, a random delay is added to each sentry node to avoid threshold checking at multiple sentry nodes at the same time. In addition, instead of the *ReliableComm*, the generic communication protocol of TinyOS, *GenericComm* is used to send the threshold flags to the gateway node so that the gateway node does not miss the flags even though it is communicating with the leaf nodes for other tasks, which are typically based on the *ReliableComm*.

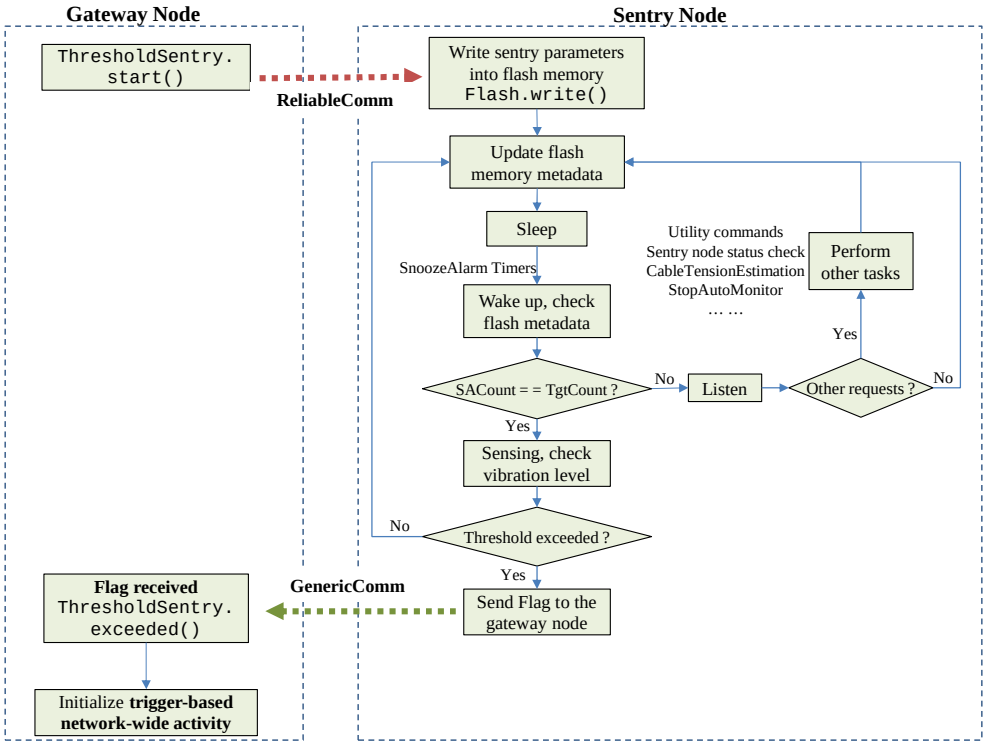


Figure 3. Block diagram of the modified ThresholdSentry service

2.2 Autonomous Monitoring

Long-term autonomous SHM with a WSSN requires a high level software manager to handle the overall network activities. The major goal of the software manager is to ensure that the tasks of the network are properly scheduled and especially the trigger-based tasks are executed without interrupting the on-going or other scheduled tasks. An application called *AutoMonitor* (Rice et al., 2010) was developed for the Imote2 smart sensor networks. *AutoMonitor* starts by the user providing an input file which is read by the gateway node. The input file includes the monitoring parameters such as the sensing parameters, *ThresholdSenry* parameters, execution period of each task, network Node IDs involved in different tasks, etc. These parameters are parsed and then used by the gateway node to control the network in an autonomous manner until the user decides to terminate *AutoMonitor* by inputting a stop command. Task scheduling is realized by assigning different timers to the scheduled tasks, which fire at different times according to the execution periods. Flags are used to mark the status of different tasks and to determine the order of execution when multiple tasks are in the queue. In addition, since all tasks involve waking up the network first, *AutoMonitor* takes advantage of the `SnoozeAlarm.awake` event, which is signaled after the `SnoozeAlarm.wakeup` command is executed and returns all the node IDs of the nodes that were successfully woken up. These woken up nodes are then included in the execution of the followed network-wide operations.

One issue with *AutoMonitor* is that the operations of the timers for scheduling tasks and the actual code for executing the tasks are all lumped in the same file, making the maintenance and expansion of *AutoMonitor* extremely difficult especially when a large number of tasks are involved. For example, all execution of tasks is implemented in the single `SnoozeAlarm.awake` event as mentioned before. Therefore, many different flags need to be carefully checked before a task can be executed. A better design would be separating the task scheduling (timer) component from the task execution component.

To make the *AutoMonitor* easier to maintain and expand, a new service module called *AutoScheduler* is developed which takes care of the scheduling timers and the initiation of task execution after the timers are fired. In the *AutoScheduler* module, a parametric interface *ScheduleApp* provides a command *ScheduleApp.registerApp* to register different network-wide tasks and a command *ScheduleApp.executeApp* to execute these tasks. The registration involves assigning a scheduling timer to each task, defining the name of the task, execution period, priority, and the name of the function which will be executed once the command *ScheduleApp.executeApp* is called. Meanwhile, another interface called *AutoScheduler* provides two commands, *AutoScheduler.start* and *AutoScheduler.stop*, to start and stop the scheduling timers. *AutoScheduler* provides a self-contained control center for

scheduling all network tasks and can avoid any potential conflict between tasks in order to facilitate smooth operation of autonomous monitoring.

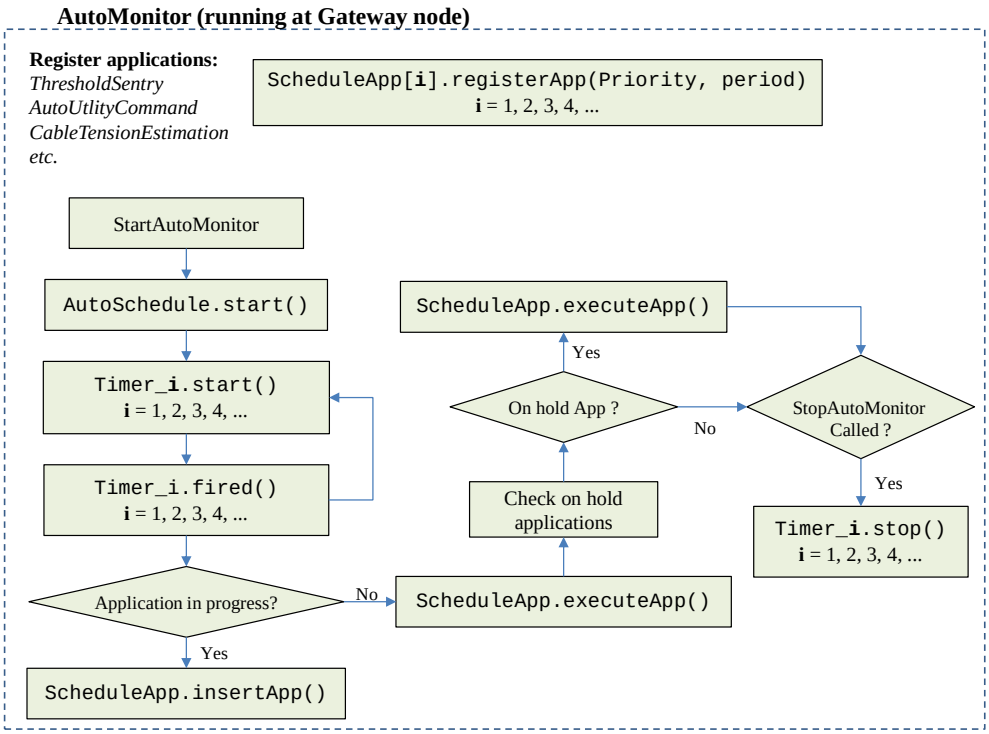


Figure 4. Block diagram of the new *AutoMonitor* and embedded *AutoScheduler*

As shown in Fig. 4, the *AutoMonitor* application first registers all tasks by calling the *ScheduleApp.registerApp* command. Once the *StartAutoMonitor* command is called, all registered scheduling timers within *AutoScheduler* are started. When a timer is fired, *AutoScheduler* first checks whether there is an on-going task or not. If there is an on-going task in the network, *ScheduleApp.insertApp* is called to insert this scheduled task into the task queue and put the task on hold. If not, *AutoScheduler* executes the task immediately. Once the execution is finished, it checks the current task queue and executes the one with the highest priority if the queue is not empty. *ScheduleApp.insertApp* is also applied to deal with the new *ThresholdSentry* implementation, in which a sentry flag can arrive at a random time at the gateway node. When the sentry flag is received by the gateway node, the trigger-based task is either executed immediately if no on-going task or inserted into the task queue otherwise. All timers are started right after they are fired to schedule periodic tasks for continuous monitoring. *AutoMonitor* can be stopped by calling the *StopAutoMonitor* command which sends a flag to *AutoScheduler*. After the execution of the immediate task, all timers are stopped to terminate the whole monitoring process.

2.3 Full-scale implementation

The 2nd Jindo Bridge is one of the twin cable-stayed bridges located in the southwestern tip of Korean Peninsula, linking the Jindo Island to Korean Peninsula (see Fig. 5). It is a three span steel-box girder cable-stayed bridge with a 344 m main span and two 70 m side spans. The steel-box girder is supported by sixty stay cables connected to two A-shaped steel pylons supported by concrete piers. Fig. 5 shows the picture of both the 1st (right) and the 2nd (left) Jindo Bridges.

Two major deployments of wireless SHM system have been implemented on the 2nd Jindo Bridge between the year 2009 and 2011 with the goal of realizing a large-scale long-term autonomous monitoring system for civil infrastructures using WSSN. The first WSSN was deployed in 2009 with 70 sensor nodes, each is composed of Imote2 and SHM-A sensor board. The entire network was divided into two sub-networks with two separate base stations. Most nodes were powered by D-cell batteries while a few nodes were equipped with rechargeable batteries and solar panels. Both sub-networks use single-hop communication. More details of the first deployment can be found in Jang et al. (2010). In the second deployment in 2010, both hardware and software were significantly improved. The number of sensor nodes was increased to 113, with total 669 sensor channels. The network was divided into four sub-networks. All sensor nodes were powered by solar panels in this deployment. Multi-hop communication protocol was

implemented for one of the cable sub-networks and improved fault tolerance features were added. More details can be found in Jo (2012).

In the 2010 deployment, the new *AutoMonitor* with the *AutoScheduler* was added to the SHM system. After this deployment, a number of further improvements and upgrades were made in 2011 including the new design of *ThresholdSentry* described in Section 2.1. Fig. 6 depicts the layout of sensor nodes in 2011 and the detailed parameters of each sub-network are given in Table 1. After the improvement in 2011, stable performance of the networks were reported (Jo, 2012) during a monitoring period of six months from October, 2011 to March, 2012.

Figure 5. The first (right) and the second (left) Jindo Bridges

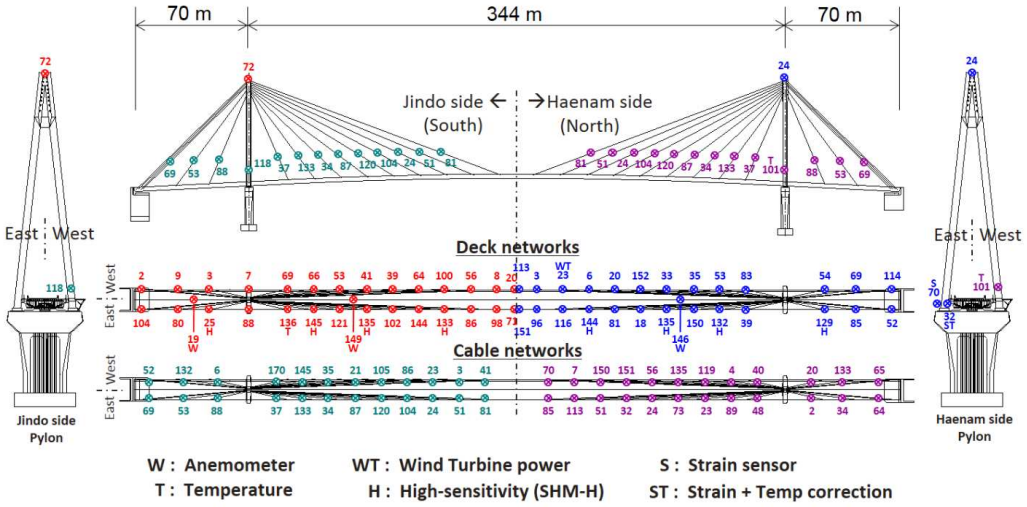


Figure 2. Wireless smart sensors deployed on the 2nd Jindo Bridge in 2011

Table 1. Main parameters of the 2011 Jindo Bridge deployment

		Haenam side		Jindo side	
		Deck	Cable	Deck	Cable
Communication channel		Ch 25	Ch 15	Ch 26	Ch 20
Network size		30 nodes	26 nodes	31 nodes	26 nodes
Sentry node		3 nodes	2 nodes	3 nodes	2 nodes
Functionalities	Common	*AutoMonitor, RemoteSensing, AutoUtilsCommand, ChargerControl, ThresholdSentry, and SnoozeAlarm*			
	Optional	*DDA*	*CTE*	*DDA*	*CTE*
Communication protocol		Single-hop	Single-hop	Single-hop	Multi-hop

IV. SHORT-TERM (BASE-STATION FREE) SHM WITH WSSNS

The main difference between the long-term and the short-term SHM systems proposed in this paper is that the latter one runs without a base station. As a result, special considerations are needed in different aspect of the autonomous network operation, such as task scheduling, sensing triggering, data storage, and energy consumption. As an on-going effort, the proposed implementation strategies are discussed in this section. Table 2 also summarizes the comparison between the autonomous sensing schemes with and without a base station.

4.1 Task Scheduling

In the long-term SHM system, the gateway node is assumed to stay awake all the time during the deployment so that *AutoScheduler* runs continuously on the gateway node. Even though the gateway node may reset due to unexpected reasons, the capability of recovering to the state before the resetting was added to ensure robust and continuous operation. However, in the case of the short-term, base-station free wireless SHM system, all nodes are treated equally in the sense that they all have to conserve energy by spending most of their time in deep sleep mode while waking up periodically to listen to external command. There is no gateway node to run these independent timers in a continuous manner to schedule tasks. Instead, a leaf node can be designated as the scheduler node. The scheduling parameters such as the execution periods of tasks and priority can be stored in the flash memory of the scheduler node. When the scheduler node wakes up, it checks the amount of time that has passed since the start of the timer by counting the number of sleep cycles, similar to the self-scheduled sentry nodes in the new *ThresholdSentry* described in Section 2.1. The task is then executed or put in the queue if the execution period has been exceeded.

4.2 Sensing triggering

In the long-term SHM WSSN, the sentry node which detects significant structural vibration always sends a triggering message to the gateway node. The latter one then initiates network sensing. In the case of base-station free WSSN, however, sentry nodes can initiate network sensing right after detecting significant events without sending triggering messages to other nodes. There are challenges such as multiple sentry nodes initiating sensing at the same time and potential conflict between triggered sensing and other running tasks in the network. One solution is to designate only one sentry node in the network and limit the network to perform triggered sensing only.

4.3 Data storage

In wireless SHM systems with base station, monitored data such as acceleration, strain, temperature, humidity, and light are transferred to the gateway node and then stored in the base station. However, this centralized data storage method cannot be applied to base-station free WSSN because the memory space on one sensor node is not able to store the data collected by the entire sensor network. Therefore, a decentralized data storage mechanism is required in which data acquired by each leaf node has to be stored locally. The data can then be harvested after the sensor nodes are retrieved from the deployment.

One issue associated with decentralized data storage is that a common identifier is needed for each local data set so that the data sets collected from the same event can be recognized when harvested. In addition, the storage space on each sensor node is limited and can only store a limited number of data sets; therefore, early stored data sets may have to be overwritten by more significant sets collected later. The identifier can also be used to make sure that the sensor nodes in the network are keeping/erasing data sets from the

same event. To provide the common identifier to each sensor, the node initiating sensing can generate a random number and send it to each leaf node before sensing begins.

4.4 Energy consumption

Wireless sensors are usually battery-powered and have limited power supply. Therefore, energy harvesting and power saving are two crucial strategies to ensure the designed operating life of the WSSN. Solar energy harvesting adopted by the long-term SHM systems will continue to be used to support the operation of the short-term base-station free wireless SHM system. Common power-saving strategies including duty-cycling and energy-aware operation will also be implemented in the proposed system.

In addition, optimization will be carried out for individual functions to reduce energy consumption based on the characteristics of the base-station free system. For example, in the long-term wireless SHM system with base station, sensing parameters are distributed by the gateway node when initializing network sensing. To reduce communication overhead and hence reduce energy consumption, these sensing parameters can be stored locally in each leaf node. During the autonomous operation of the sensor network, when network sensing is initialized, there is no need to broadcast sensing parameters every time; hence communication overhead and energy consumption can be reduced.

Table 2. Comparison between autonomous sensing scheme with and without base station

Functionality	AutoMonitor with Base Station	Base-Station-Free AutoMonitor
Task scheduling	Use *AutoScheduler* to schedule all tasks by the Gateway node	Designate a scheduler node and store scheduling parameters into the flash memory
Threshold Triggering	Sentry node sends triggering message to the gateway node, which then initializes network sensing	Sentry node initializes network sensing
Data Storage	All data sets are transferred back to and stored in the base station	Data sets are stored locally at each sensor node before retrieved back. Unique identifiers are needed for each set of data.
Energy consumption	Duty-cycling, energy-aware operation	Duty-cycling, energy-aware operation, further optimization

V. CONCLUSION

Autonomous operation is essential to a successful SHM system. The abilities of wireless communication and on-board computation of wireless smart sensors offer great possibility and flexibility in the design of multi-functional SHM systems, but pose challenges in the realization of fully autonomous operation. In this paper, both long-term and short-term autonomous SHM Systems are proposed. The long-term SHM system aims for a deployment which lasts between a few months and several years. These systems continuously monitor the long-term behavior of civil structures throughout its service life and provide key information to guide structural maintenance and operation, emergency response and planning, etc. Focus in this paper has been placed on the new strategies integrated into the software design of threshold triggering mechanism and autonomous monitoring manager based on the lessons learned from the previous full scale deployments. By switching the scheduling of the *ThresholdSentry* events from the gateway node to the sentry nodes themselves, unnecessary communication between the gateway and sentry nodes is eliminated and the power drainage issue with multi-hop network is addressed. The introduction of the scheduling service, *AutoScheduler*, makes easier the maintenance and expansion of the *AutoMonitor* application and provides smooth operation of WSSNs with complex functions.

On the other hand, the short-term SHM system is designed to run for a few days to weeks without a central base station in order to create an easy-to-deploy and efficient autonomous system for campaign monitoring. Different strategies have been discussed to address the issues in task scheduling, threshold triggering, data storage, and energy consumption. Such a system can benefit a wide range of civil structures especially for those located at remote areas where external power supply for supporting a base station is unavailable.

REFERENCES

[1] Rice, J.A., Mechitov, K.A., Spencer Jr., B. F., and Agha, G.A. (2010), "Autonomous smart sensor network for full-scale structural health monitoring." *Proc. SPIE Smart Structures/NDE 2010*, San Diego, CA.

[2] Rice, J. A. and Spencer Jr., B. F. (2009), "Flexible Smart Sensor Framework for Autonomous Full-scale Structural Health Monitoring." *NSEL Report Series*, No. 18, University of Illinois at Urbana-Champaign.

[3] Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. and Culler, D. (2003), "The nesC language: a holistic approach to networked embedded systems." *Proceedings of the ACM SUGPLAN Conference on Programming Language Design and Implementation, ser. PLDI'03*. New York, NY, USA: ACM, 2003, pp. 1-11.

[4] Sim, S. H. and Spencer Jr., B. F. (2009). "Decentralized Strategies for Monitoring Structures using Wireless Smart Sensor Networks." *NSEL Report Series*, No. 19, University of Illinois at Urbana-Champaign.

[5] Mechitov, K. A. (2011). A Service-oriented Architecture for Dynamic Macroprogramming of Sensor Networks. Ph.D. Thesis, University of Illinois at Urbana-Champaign.

[6] Rice, J. A., Mechitov, K., Sim, S. H., Nagayama, T., Jang, S., Kim, R., Spencer Jr., B. F., Agha, G. and Fujino, Y. (2010), "Flexible Smart Sensor Framework for Autonomous Structural Health Monitoring," *Smart Stuct. Syst*. 6, 423-38.

[7] Jo, H., Sim, S. H., Mechitov, K.A., Kim, R., Li, J., Moinzadeh, P., Spencer Jr, B. F. et al. (2011), "Hybrid wireless smart sensor network for full-scale structural health monitoring of a cable-stayed bridge." *Proc. SPIE Smart Structures/NDE 2011*, San Diego, CA.

[8] Nagayama, T., Moinzadeh, P., Mechitov, K., Ushita, M., Makihata, N., Ieiri, M., Agha, G., Spencer Jr., B. F., Fujino, Y., and Seo, J.-W. (2010). "Reliable multi-hop communication for structural health monitoring." *Smart Stuct. Syst*. 6(5):481-504.

[9] Jang, S., Jo, H., Cho, S., Mechitov, K., Rice, J.A., Sim, S.H., Jung, H.J., Yun, C.B., Spencer, Jr., B.F. and Agha, G. (2010), "Structural Health Monitoring of a Cable-stayed Bridge Using Smart Sensor Technology: Deployment and Evaluation," *Smart Stuct. Syst*. 6, 439-459.

[10] Jo, H. (2013). Multi-scale Full Structural Health Monitoring using Wireless Smart Sensors. Ph.D. Thesis, University of Illinois at Urbana-Champaign.