Smart Structures and Systems, *Vol. 18, No. 5 (2016) 885-909*
DOI: http://dx.doi.org/10.12989/sss.2016.18.5.885

Synchronized sensing for wireless monitoring of large structures

Robin E. Kim*[1], Jian Li[2], Billie F. Spencer, Jr.[3], Tomonori Nagayama[4] and Kirill A. Mechitov[5]

[1]Fire Research Institute, Korea Institute of Civil engineering and building Technology
Fire Research Institute, 64, Mado-ro 182 beon-gil, Mado-myeon Hwaseong-si, Gyeonggi-do 18544 South Korea
[2]Department of Civil, Environmental, and Architectural Engineering, The University of Kansas, Lawrence, KS 66045, USA
[3]Department of Civil and Environmental Engineering, University Illinois Urbana-Champaign, Urbana, IL 61802, USA
[4]Department of Civil Engineering, The University of Tokyo, Tokyo 113-8656, Japan
[5]Department of Computer Science, University Illinois Urbana-Champaign, Urbana, IL 61802, USA

(Received December 8, 2015, Revised June 30, 2016, Accepted July 18, 2016)

Abstract. Advances in low-cost wireless sensing have made instrumentation of large civil infrastructure systems with dense arrays of wireless sensors possible. A critical issue with regard to effective use of the information harvested from these sensors is synchronized sensing. Although a number of synchronization methods have been developed, most provide only clock synchronization. Synchronized sensing requires not only clock synchronization among wireless nodes, but also synchronization of the data. Existing synchronization protocols are generally limited to networks of modest size in which all sensor nodes are within a limited distance from a central base station. The scale of civil infrastructure is often too large to be covered by a single wireless sensor network. Multiple independent networks have been installed, and post-facto synchronization schemes have been developed and applied with some success. In this paper, we present a new approach to achieving synchronized sensing among multiple networks using the Pulse-Per-Second signals from low-cost GPS receivers. The method is implemented and verified on the Imote2 sensor platform using TinyOS to achieve 50 μs synchronization accuracy of the measured data for multiple networks. These results demonstrate that the proposed approach is highly-scalable, realizing precise synchronized sensing that is necessary for effective structural health monitoring.

Keywords: full-scale railroad bridge implementation; low-cost GPS receiver; multiple networks; synchronized sensing; PPS signal; wireless smart sensors

1. Introduction

Wireless sensor technology promise to improve our ability to monitor the structural integrity of civil infrastructure in real time, which is often large in dimension and extremely complex. Traditional wired sensors limit their potential for such applications due to the associated high cost of equipment, labor-intensive installation of the cabling plant, and data inundation due to

∗Corresponding author, Research Specialist, E-mail: robineunjukim@kict.re.kr centralized data collection (Rice and Spencer 2009). Wireless smart sensors (WSSs) offer the possibility of addressing these limitations. Recent advance in micro-electro-mechanical systems (MEMS) technology has reduced the cost of sensors. Wireless communication has eliminated the need for much of the cabling. The on-board computational capability of WSSs also reduced the risk of data inundation by extracting the important features of the data, only transmitting essential information to the central base station. Characteristics such as small size and self-powered have led to a few successful long-term monitoring applications (Lynch, Wang *et al*. 2005, 2006b, Pakzad, Fenves *et al*. 2008, Jang, Jo *et al*. 2010, Cho, Jo *et al*. 2010, Jo, Sim *et al*. 2012). While WSSs have seen great advancement and offer tremendous opportunities, several challenges remain.

One of the critical issues in WSSs for Structural Health Monitoring (SHM) is providing synchronized sensing. In wireless sensor networks, each WSS operates based on its own clock; the WSS network lacks a centralized time reference, and its performance is susceptible to the quality of the clock’s oscillator (e.g., offset, jitter, frequency error and stability; Windl, Dalton *et al.* 2006). Nagayama, Spencer *et al*. (2007) investigated the effect of synchronized sensing error on the accuracy of modal analysis. Because synchronization error causes substantial phase shift in mode shapes, inaccurate synchronized sensing protocol may lead to false positive or false negative results in damage detection. Thus, for successful use of WSSs for SHM, an accurate synchronized sensing protocol which is able to minimize the error down to an acceptable level must be used (Spencer *et al*. 2008).

Several researchers have proposed time synchronization schemes for WSS networks. These protocols can be categorized as software-based and hardware-based approaches. A typical software-based protocol can time-synchronize the network by periodically sending beacon signals to estimate clock offset of the nodes. (Lynch 2007). Hardware-based time synchronization schemes provide communication-free time synchronization among WSSs. Such schemes typically consist of hardware interrupts for each WSS node to adjust the internal clock oscillator (Chen, Wang *et al*. 2011). These hardware-based solutions can be costly, due to the need for interrupt hardware on each node. In addition, synchronization of the network depends on the precision of the selected hardware (Elson and Römer 2003). In general, clocks on wireless nodes can be synchronized to a high level of precision. However, synchronized clocks do not necessarily yield synchronized data.

Traditional synchronization approaches aim to share the reference clock time with all remote sensor nodes. Such protocols cannot control sensing tasks as in a wired network (Nagayama, Spencer *et al.* 2008). The reason lies in the fact that, although the clocks are accurately synchronized, random differences are present in the time at which data collection begins and in the realized sampling rate; non-real time operating system (OS) and non-negligible individual differences in ADC clock frequency, which are typical on WSSs, introduce the random differences unless appropriately designed. To utilize the measured data for civil infrastructure monitoring, these differences must be eliminated. Additionally, because of the large scale of civil engineering structures, multiple networks are often necessary to monitor the entire structure, and sensing duration can be quite long, requiring measurements to remain synchronized over extended periods of time.

In this paper, we propose a new synchronization strategy, which can achieve high-accuracy synchronized sensing in multiple WSS networks for extended sensing duration. The proposed approach combines hardware- and software-based methods to achieve the necessary accuracy while minimizing the number of GPS receivers used within the network. The developed protocol does not utilize expensive and power-hungry Phase Lock Loop (PLL) circuits, which are often employed in traditional time synchronization approach with GPS receivers. Instead, the Coordinated Universal Time (UTC) information and the Pulse-Per-Second (PPS) signal are used along with the resampling algorithm resulting in an inexpensive and power efficient solution. Especially the PPS signals are read at the digital interface, reducing time delay associated with analog signal analysis. This protocol first utilizes a hardware-based approach to establish accurate clock synchronization among base stations of individual sub-networks according to the global time from the GPS receivers. Subsequently, a software-based approach is used to provide synchronized sensing within each sub-network, in which the hierarchical root nodes are already synchronized through the hardware-based approach. The method was implemented and verified on the Imote2 sensor platform running TinyOS, showing high-accuracy, long-term, synchronized sensing performance for an arbitrary number of sensor nodes. Finally, the prospective use of the algorithm has been tested by synchronizing two distinct subnets: one on a full-scale railroad bridge network and the other on a moving train network. These results show great potential of the proposed method in facilitating wide application of WSS networks for monitoring large-scale civil infrastructure.

2. Review of traditional time synchronization approaches

This section provides an overview of traditional software- and hardware- based clock synchronization protocols. Applications of these protocols for monitoring civil structures using wireless sensors are also reviewed.

A number of software-based protocols have been developed to achieve synchronization of the clocks on CPUs distributed in a network. An example is the Network Time Protocol (NTP). NTP synchronizes CPU clocks with the precision of few milliseconds using a hierarchical, semi-layered system of time sources (Mills 1991). The protocol may not have sufficient accuracy for applications that require precise clock synchronization and suffers from delays associated with asymmetric round-trip message delivery (Elson, Lewis *et al.* 2002). Advanced clock synchronization protocols to minimize the cost, power for communication, and delays have been introduced. Reference Broadcast Synchronization (RBS), Timing-sync Protocol for Sensor Network (TPSN), and Flooding Time Synchronization Protocol (FTSP) are examples of such protocols used for WSSs. RBS uses receiver-to-receiver clock synchronization by exchanging the clock information among nodes in the network. Thus, RBS is usually not adequate for large networks or multi-hop applications. TPSN is a typical sender-to-receiver clock synchronization using two-way communication. FTSP is similar to TPSN, which uses sender-to-sender synchronization. The protocol provides time information through the Media Access Control (MAC) layer, which eliminates error from propagation time, reduces jitter, and increases the reliability (Maróti 2004). On initialization, the sender node broadcasts a data packet, which contains time information as a format of a timestamp at the end of the packet. Upon the reception of the packet by the receivers, the nodes timestamp the local clock reading and calculate the bit offset required to reconstruct the message. The design of FTSP allows high accuracy to be obtained and easy handling of dynamic topology changes and multi-hop. The protocol can synchronize all nodes in a network to the sender node with high accuracy (Roche 2006). Easy implementation and relatively inexpensive cost to realize the protocol in software-based approaches encouraged wide adoption of the method in WSS networks.

A critical problem in software-based approaches is that the nodes are synchronized to the sender node. Any drift in the sender clock is propagated to the entire network. Moreover, multiple networks cannot easily be synchronized. In an effort to eliminate such problems, hardware-based approaches have been proposed by several researchers. One of the popular hardware-based clock synchronization protocols uses a GPS receiver. The GPS system time is referenced to UTC which is within 50 ns accuracy and is transmitted in the C/A (Coarse/Acquisition) and P(Y) messages from the satellites. However, to use the UTC information, the signals should be decoded. This process may add time delays that degrade the synchronization accuracy. Instead, most stand-alone GPS receivers are capable of outputting PPS signals when they are locked with satellites. PPS signals is an electrical signal with a width less than one second and a sharp rising edge (or falling edge) corresponds to the clock change in the satellites. The known accuracy of the PPS signals is less than 200 ns (Mannermaa 1999). The pulses can easily interrupt hardware without introducing delays. Although stand-alone type GPS receivers give accurate time information, implementing a GPS receiver on each wireless node would be expensive, both in terms of cost and power consumption.

Several of these software-based clock synchronization protocols have been implemented for structural health monitoring. Wang, Lynch *et al*. (2005) employed a rough software-based protocol on a low-power wireless sensor unit developed for SHM using commercially available embedded components. In their approach, the network is composed of leaf nodes, each of which has a gateway node to coordinate activities, including time synchronization (see Fig. 1). The leaf nodes in the network synchronize their clocks to the gateway node upon receiving a beacon signal from the gateway node. The measurement is also initialized as the signal arrives, eliminating any time delays. The protocol assured reliable communication regarding packet loss during operation of the monitoring system (Wang, Lynch *et al*. 2005, Lynch and Loh 2006, Lynch, Law *et al*. 2004, Lynch, Wang *et al*. 2006). Lynch, Wang *et al*. (2006) implemented the system on the Geumdang Bridge in Korea and validated the protocol. A network composed of 14 wireless sensor nodes was deployed to monitor the bridge, which has a total length of 273 m. Comparing the acceleration signals collected from wireless sensors to that from wired sensors, the protocol showed 5-10 ms clock synchronization accuracy among 15 sensors (Lynch, Wang *et al*. 2006). These papers did not explore the synchronization errors associated with random delays in initial start-up, nor did they consider clock drift that can dominate measurements over extended sensing period.

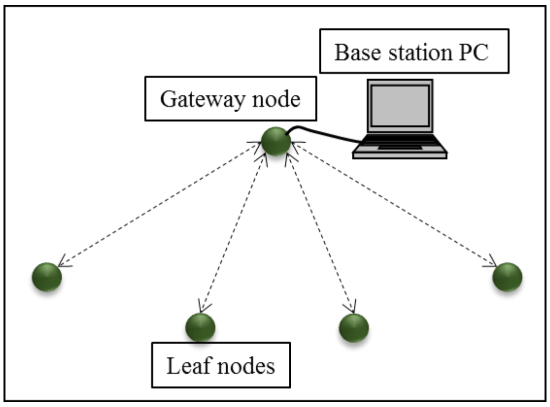


Fig. 1 An example of a WSS network topology

To overcome some limitations of the software-based methods that incorrect time information propagates to the network and hardware-based methods that establishing the network may be inefficient, Sazonov, Krishnamurthy *et al*. (2010) proposed a clock synchronization protocol combining two approaches and implemented it in their Wireless Intelligent Sensor and Actuator Network (WISAN). The gateway node periodically broadcasts beacon signals to the leaf nodes. As a leaf node receives beacon signals, the internal timer is updated to compensate time difference. When stabilized, the estimated maximum drift on each gateway node is ±30 µs per second (Sazonov, Krishnamurthy *et al*. 2010). A laboratory scale test using two GPS receivers validated that the proposed protocol achieved about 15 µs synchronization error within two separate sub-networks. A field test composed of forty-four leaf nodes on a four-steel-girder bridge located in NY, USA also validated the protocol. The maximum time error among respective leaf nodes was about 23 µs.

However, as will be shown in the following section, synchronized clocks do not necessarily yield synchronized sensing. Nonlinear drift in the gateway node clocks and the randomness in the clock speed should also be adjusted to achieve synchronized sensing in the systems.

3. SHM-specific synchronized sensing protocols

Traditional clock synchronization protocols alone do not produce synchronized data. Clock synchronization does not compensate for differences in sampling rate among leaf nodes, fluctuations in the sampling frequency, and the start time of sensing (Nagayama and Spencer 2007). For example, although the internal clocks in two leaf nodes shown in Fig. 2 were synchronized initially at t= 0.4 seconds, clock speed differences resulted in differences in sampling rates and hence unsynchronized measurements. In Fig. 3 the nodes have exactly the same sampling rate; however, sensing may not start precisely at the same time in each of the leaf nodes, resulting in the unsynchronized data shown. This section reviews two synchronized sensing protocols tailored for SHM applications that address such issues.

3.1 Synchronized sensing through resampling

To achieve synchronized sensing, Nagayama and Spencer (2007) developed a protocol that addresses problems with both clock drift and differences in the start of sensing. The proposed method uses a two-step synchronization process: (i) clock synchronization, adopting the FTSP time stamping method, and (ii) a subsequent polyphase resampling of the data to put it all on a common time scale. The clock synchronization step corrects the most common clock errors in discrete clocks, which are offset and clock drift. Clock drift occurs due to imperfect speed differences among clocks; therefore, even if clocks are perfectly synchronized at time t = 0, the drift in the clocks will result in an offset between leaf node clocks at future times. To better understand this problem, consider the measured clock offset for nine leaf nodes with respect to the gateway node shown in Fig. 4. As seen here, although the nine nodes are synchronized with the gateway node at t=0, as time progresses, the offset between the clocks increases linearly.

In this approach, prior to the start of sensing, both clock offset and clock drift rate with relative to the gateway node are estimated for each leaf node, as illustrated in Fig. 5. Before the start of sensing, the gateway node, regularly broadcasts beacon signals for a period of 30 seconds. Each leaf node records the local time at which the signals were received, and estimates the offset for the leaf node clock relative to the gateway's clock. When the gateway node finishes sending beacon signals, the leaf nodes estimate their drift rate with respect to the gateway node based on the collected offset values using linear regression.

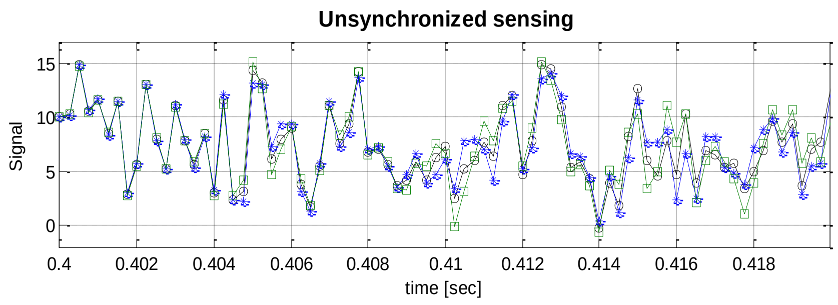


Fig. 2 Illustration of unsynchronized data resulting from differences in sampling rate

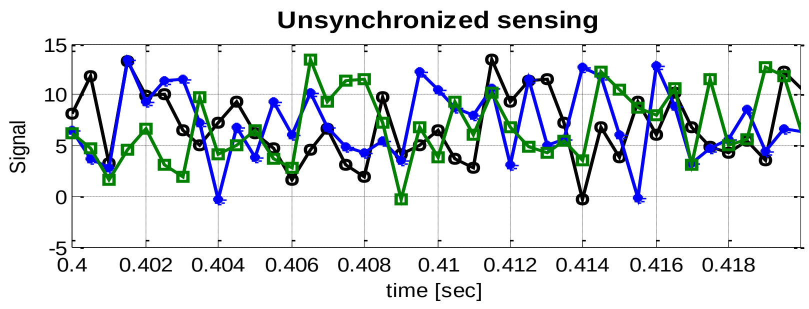


Fig. 3 Illustration of unsynchronized data resulting from differences in sensing start time

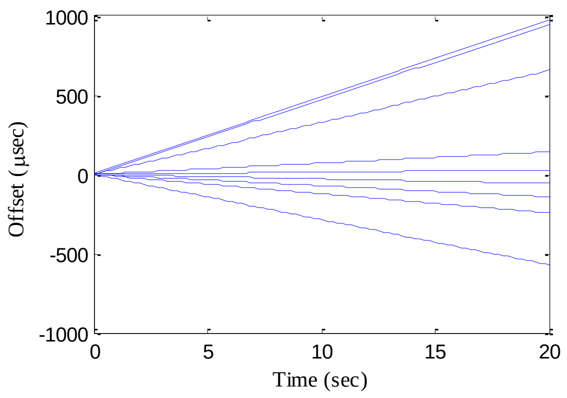


Fig. 4 Clock drift in Imote2s (Nagayama and Spencer, 2007)

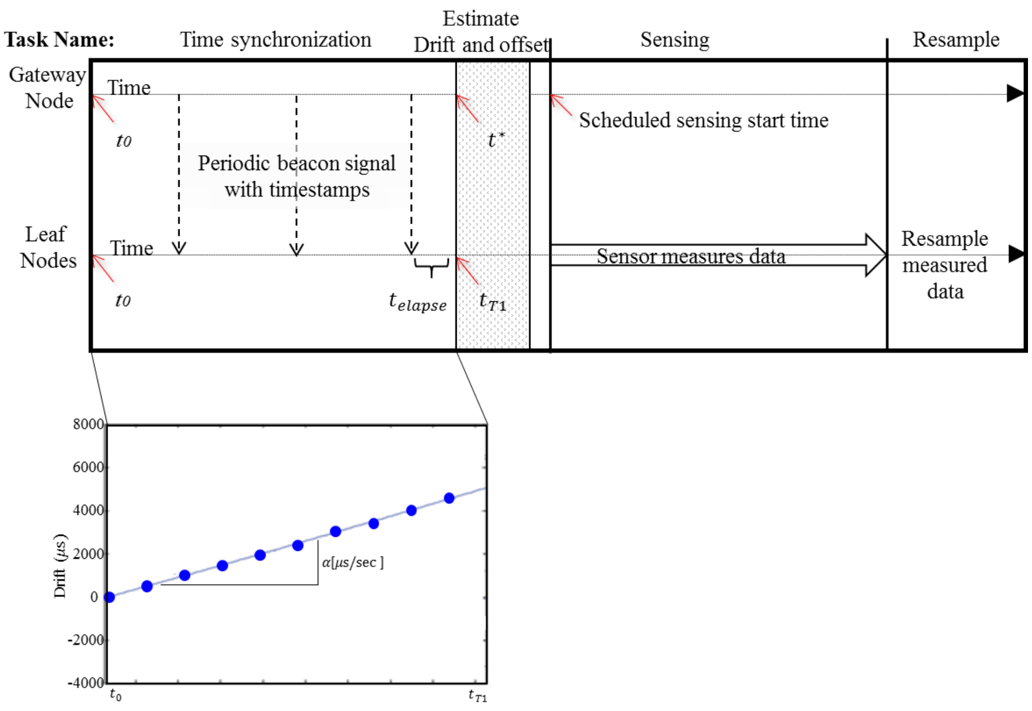


Fig. 5 Synchronized sensing scheme with linear clock drift estimation and resampling: gateway node broadcast periodic beacon signals (top box, top line); leaf nodes store received time stamp to estimate drift and offset and start sensing (top box, bottom line); example of drift estimation in a leaf node during the time synchronization (bottom box)

Once the synchronization process has finished, data can be collected from the sensors connected to the leaf nodes. After completion of sensing, poly-phase resampling takes place (Fig. 5). Based on laboratory tests, the estimated synchronization error of measured data for a small network is about 50 μs (Nagayama and Spencer 2007). The protocol was then implemented on a full-scale structure, the Jindo Bridge in Korea. The deployment constitutes the world's largest sensor network of its kind with 113 nodes and 669 multi-metric sensing channels. The entire network was divided into four sub-networks using two base station PCs and four gateway nodes. The data from the 4 sub-networks was synchronized in a post-facto manner after being collected on a single computer by comparing the measured responses at nodes in common to the networks. Several limitations were encountered in this approach. First, the 30 seconds required to estimate clock drift rate made capturing transient responses difficult. Additionally, the assumption that the clock drift was linear was found to be invalid, particularly for long acquisition times.

3.2 Compensation for nonlinear clock drift

In examining the data collected by the wireless sensors for an extended period of time, the assumption of linear clock drift was found to be invalid. For example, Fig. 6 shows clock offset of 4 leaf nodes, along with the measured temperature of the sensor. As can be seen in the figure, nonlinear clock drift occurs, which is highly correlated to temperature variation of the wireless node. These temperature changes can be attributed to issues such as internal heat generated by the sensor node due to sensing, external heating (e.g., by direct sunlight) etc. Fig. 7 shows Node 69, along with the linear drift rate estimated using 30 seconds of data at the beginning of sensing. As can be seen here, after collecting data for 10 minutes, the linearly corrected time on Node 69 would be off by more than 9 ms from the gateway node.

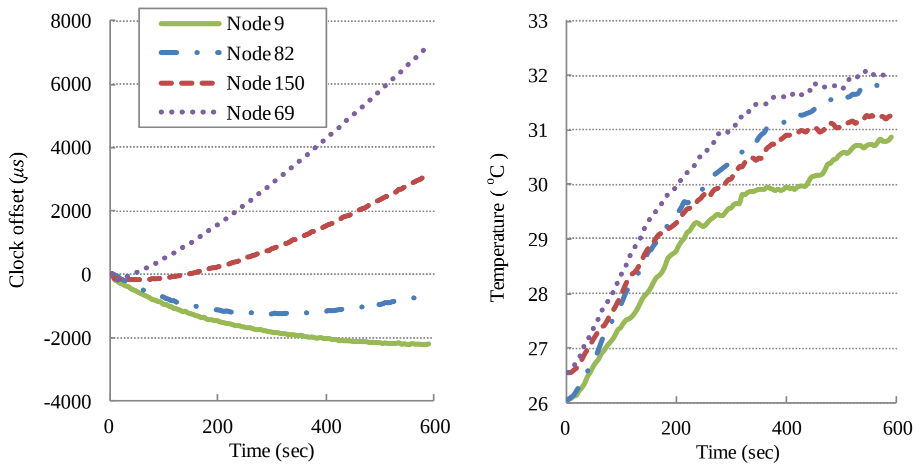


Fig. 6 Nonlinear clock drift (Li *et al*. 2015)

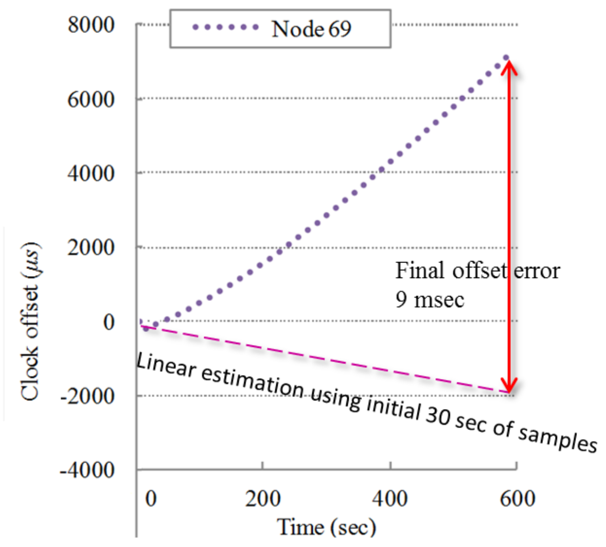


Fig. 7 Clock offset error due to nonlinear clock drift

To address the problem of nonlinear clock drift, Li, Nagayama *et al*. (2015) developed a synchronization protocol that enables synchronized sensing over an arbitrary length of sensing duration. On initializing the process, a gateway node broadcasts to the network a single beacon signal with the global timestamp. Upon reception of the timestamp, leaf nodes calculate the offset and adjust their clocks to synchronize the clocks of the network. Subsequently, the gateway node sends out the sensing start time to the leaf nodes assuming that the clocks do not drift much during the short delay before sensing starts. During sensing, the gateway node periodically broadcasts beacon signals with global timestamps, which is stored in the leaf nodes upon reception. When sensing is finished, timestamps of data are corrected using the nonlinear clock drift estimated with the beacons collected during sensing. Resampling is then performed based on the drift-compensated timestamps, which ensures synchronized measurements among the leaf nodes. Because the leaf node is simultaneously performing sample acquisition and receiving beacon signals from the gateway, conflict between these two tasks may occur, resulting in outliers in the timestamps associated with reception of the beacon signals. The protocol detects and removes these outliers prior to resampling. The proposed protocol achieved synchronized sensing with less than 50 µs synchronization error with gateway clock for long-term data collection in laboratory tests. The protocol also reduced the delay before the start of sensing by eliminating the need to estimate the leaf node's clock drift rate before sensing. However, the application of the protocol is limited to a single network. In addition, the inaccuracy of the gateway clock, which can occur due to temperature changes in the node, is propagated to the leaf nodes. Thus, a unified protocol to address all of the important issues simultaneously is needed.

4. GPS-based synchronized sensing for multiple WSS Networks

An efficient protocol that combines the advantages from both hardware-based and software-based approaches is proposed to achieve synchronized sensing. A low-cost GPS used to ensure correct timing on the gateway is combined with the resampling schemes proposed by Nagayama, Spencer *et al*. (2007) and the nonlinear drift compensation strategy proposed by Li, Nagayama *et al.* (2015) is employed. This section describes the background and implementation of this combined protocol.

4.1 Overview of the proposed synchronized sensing protocol

The proposed protocol aims to achieve the following main goals by combining both hardware- and software-based approaches: i) correct gateway clock errors using a low-cost GPS receiver, and ii) provide synchronized sensing within a sub-network by compensating nonlinear internal clock drift. The hardware-based approach enables maintaining the gateway nodes clock highly accurate among sub-networks. Such a scheme provides scalability by removing communication overhead among sub-networks. In other words, increasing the number of sub-networks does not degrade the performance of synchronized sensing. Moreover, hardware interruption can be handled asynchronously while a gateway node is performing other scheduled tasks; therefore, no delay will be added. Then, the gateway node broadcasts corrected timestamps to its network in beacon signals while leaf nodes perform a sensing task. Leaf nodes, on receiving the signals, store the local clock time and offset from reference clock (Li, Nagayama *et al*. 2015). Once measurement is done, the measured data is resampled (Nagayama and Spencer 2007). This approach removes nonlinear clock drift of leaf nodes as well as the gateway node to achieve highly precise synchronized sensing within each sub-network.

Fig. 8 summarizes the scheme of the proposed synchronized sensing algorithm. The approach achieves efficiency for multiple networks by removing the communication between different networks, while preserving high accuracy within a network. The gateway nodes are synchronized with GPS signals assuring similar degree of synchronization throughout the networks. Scalability is secured with reduced size of networks.

4.2 GPS-based synchronized sensing: hardware setup for the Imote2

A low-cost GPS receiver, GlobalTop GMS-u1LP, is chosen to provide UTC time to networks and to output PPS signals (Fig. 9(a)). The chipset on the receiver is MediaTek GPS MT3329 solution that supports up to 66 channels of satellite searching (GlobalTop 2010). The typical power consumption is only 24 mA when tracking satellites and 30 mA during acquisition (GlobalTop 2010). Also, the receiver is relatively small in size (16×16×6 mm) and inexpensive ($20 USD / unit). With those features, the selected GPS module is both cost and energy efficient and is therefore suitable for WSN applications.

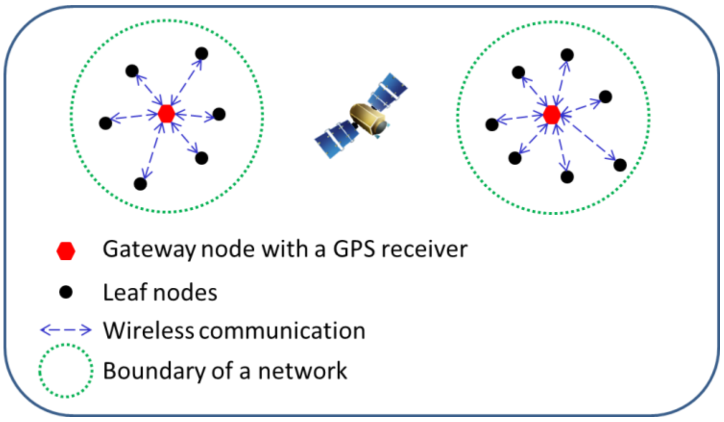
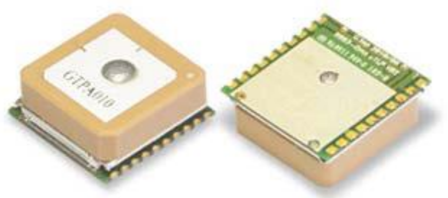
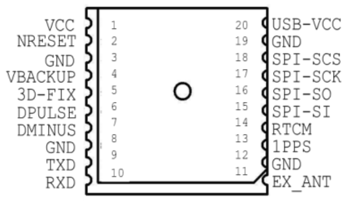


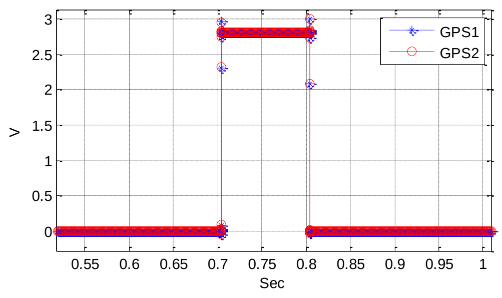
Fig. 8 Illustration of the proposed synchronized sensing approach


(a) GPS module

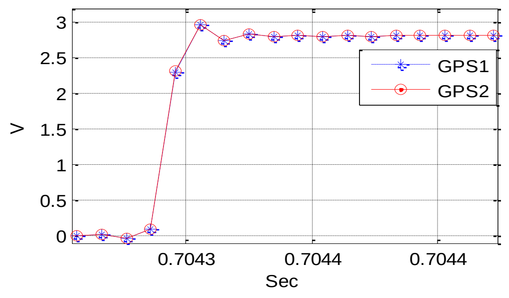


(b) Receiver pin specification

Fig. 9 GlobalTop GMS-u1LP GPS receiver

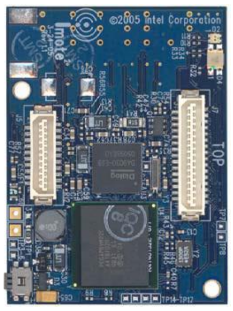


(a) sampled at 102400 Hz

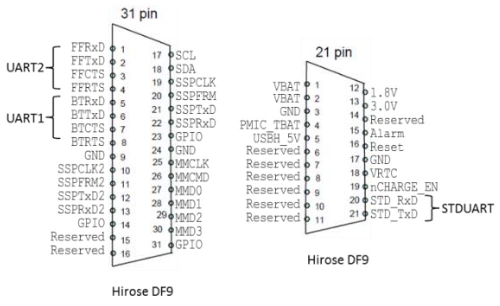


(b) zoomed PPS rise

Fig. 10 A typical GPS pulse (Kim *et al.* 2012)



(a) top view



(b) connector pin specification

Fig. 11 The Imote2 platform

To verify the applicability of the selected GPS receivers, the PPS pins, as shown in Fig. 9(b), two receivers are connected to a VibPilot (www.mpihome.com 2010) and data is collected at a high frequency (112,400 Hz). As shown in Fig. 10, when the two GPS receivers are in locked status, the PPS signals measured from them are identical, which provides the basis for nearly perfect clock synchronization if the gateway nodes of two sub-networks are connected to them (Kim, Nagayama *et al.* 2012).

Hardware pin connections from a GPS receiver to an Imote2 enabled the use of PPS and serial data output directly from the Imote2 platform. The PPS pin is connected to one of the General Purpose Input Output (GPIO) pin on an Imote2 (see Figs. 11(a) and 11(b)). The serial data output for NMEA sentence with UART Transistor-transistor Logic (TTL) communication, and thus a TXD port on the GPS receiver (Fig. 9(b)) is connected to an UART RX pin on the Imote2 (Fig. 11(b))

Hardware interrupts through PPS signals control the entire algorithm to ensure that subsequent tasks are executed concurrently on gateway nodes when a network consists of multiple sub-networks. Each PPS signal passing the GPIO register performs tasks depending on the status of the algorithm at which the hardware is interrupted (see Fig. 12). For example, 1) upon initialization of the protocol, the GPIO interrupt handler is enabled to capture the first PPS signal. Once the signal is successfully captured, the handler is disabled until the following tasks are done to ignore the subsequent PPS signals. 2) while the handler is disabled, UART port parses NMEA sentence and gets UTC information. Baud rate for the port is set 9600 bps (bits per seconds) to match with supporting speed of the GPS receiver. The port parses data every byte and closes and clears the buffer after reading the first valid UTC information. 3) Once the UTC time is obtained, the GPIO interrupt handler is re-enabled and the following PPS signals periodically adjust the Imote2 internal clock to maintain the GPS time throughout the processes. Although the standard TinyOS scheduler follows a FIFO (First In, First Out) rule, the variation of the processing time of PPS signals can be ignored because the handlers are signalled asynchronously by hardware and no other tasks are asynchronously scheduled (Gay, Levis *et al.* 2003). Thus, when PPS signal interrupt occurs, the scheduled event will follow immediately.

4.3 GPS-based synchronized sensing: software development

This section describes software development to achieve synchronized sensing The proposed algorithm implemented in PPS_based_RemoteSensing[*], a stand-alone application which aims to provide accurate synchronized sensing for multiple networks using PPS signals from GPS receivers. The application is composed of three main processes: (i) the clock synchronization of the gateway nodes based on PPS and UTC time, (ii) the synchronization of the clocks of leaf nodes, and (iii) data synchronization of leaf nodes within each sub-network. Because the PPS signal is globally synchronized with satellites, the application does not require any signal exchange among networks, hence ensuring the scalability of the proposed approach. At the same time, the communication overhead is kept the same as what is required for single network synchronization. The remainder of this section first describes the general approach needed to achieve synchronized sensing in multiple networks and then describes the specific details of the implementation on Imote2s using TinyOS.

4.3.1 Gateway Nodes: Clock synchronization in multiple sub-networks

Fig. 13 illustrates the proposed approach used to clock synchronize multiple gateway nodes. As those gateway nodes commands the network to prepare start sensing, the gateway nodes enable the GPIO pin on the Imote2. Both PPS signals and UTC time information are read until the input time is reached. At the desired UTC time (15:00:00 in the example), multiple gateway node reset their local clock (and timestamp) to zero. Then, gateway nodes calculate the sensing start time and broadcast to the subnet, such that the leaf nodes in multiple subnets initiate sensing at the same time as will be appeared in the next subsection. The proposed scheme assures that multiple subnets share a common clock time to synchronize clocks without having to exchange among subnets.

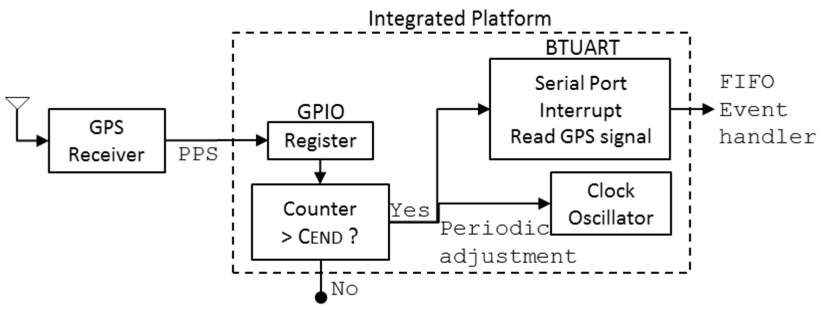


Fig. 12 Hardware interrupt scheme

[*] An Calibri font refer to the name of middleware service embedded in TinyOS.

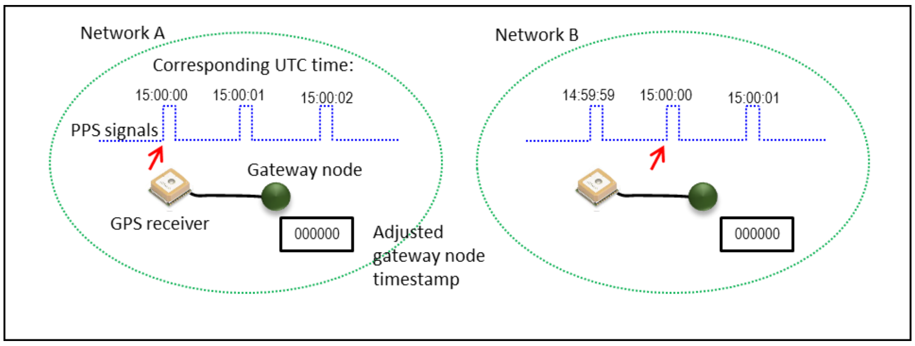


Fig. 13 Gateway node clock synchronization on the embedded application

4.3.2 Leaf nodes: Clock and data synchronization in multiple sub-networks

Before sensing starts, the clocks on leaf nodes are synchronized based on the clock information broadcasted from the gateway node (see Fig. 14). Then, the clock offsets between the leaf node and the gateway node are stored on the leaf nodes. Once the clock is synchronized, gateway node broadcast a message with wait time to start measurements. Based on the stored offset, leaf nodes calculate the timestamp at which each should start sensing. Once sensing is finished, resampling takes place to achieve data synchronization by eliminating the initial start time error and randomness in the leaf node's clock speeds. Because the clocks of all gateway nodes are synchronized initially, the resultant data after resampling is expected to have the same starting timestamp among leaf nodes in different sub-networks.

4.3.3 Implementation on Imotes

Implementation of the algorithms on Imote2s to achieve synchronized sensing for multiple sub-networks is non-trivial due to random delays in hardware operation, clock oscillator speed uncertainty, task scheduling in TinyOS, etc. As a result, although the clocks of the gateway nodes are synchronized initially, this synchronization cannot be guaranteed over extended periods of time. Any errors in synchronization on the gateway nodes will be propagated subsequently to the leaf nodes, making collection of synchronized data impossible. An application PPS_based_RemoteSensing is developed to address these problems using the PPS signal from the GPS.

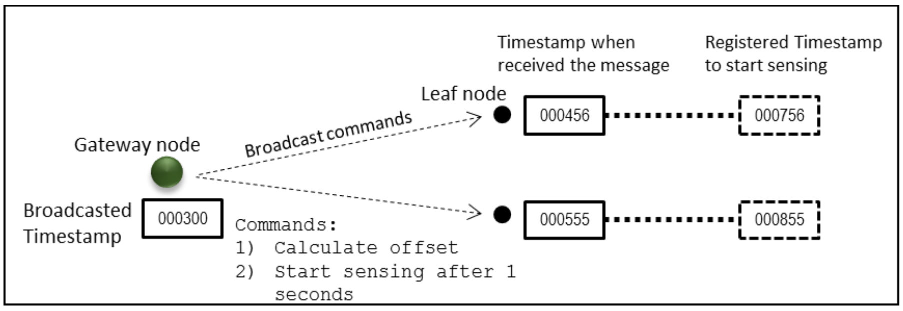


Fig. 14 Leaf nodes clock synchronization on the embedded application

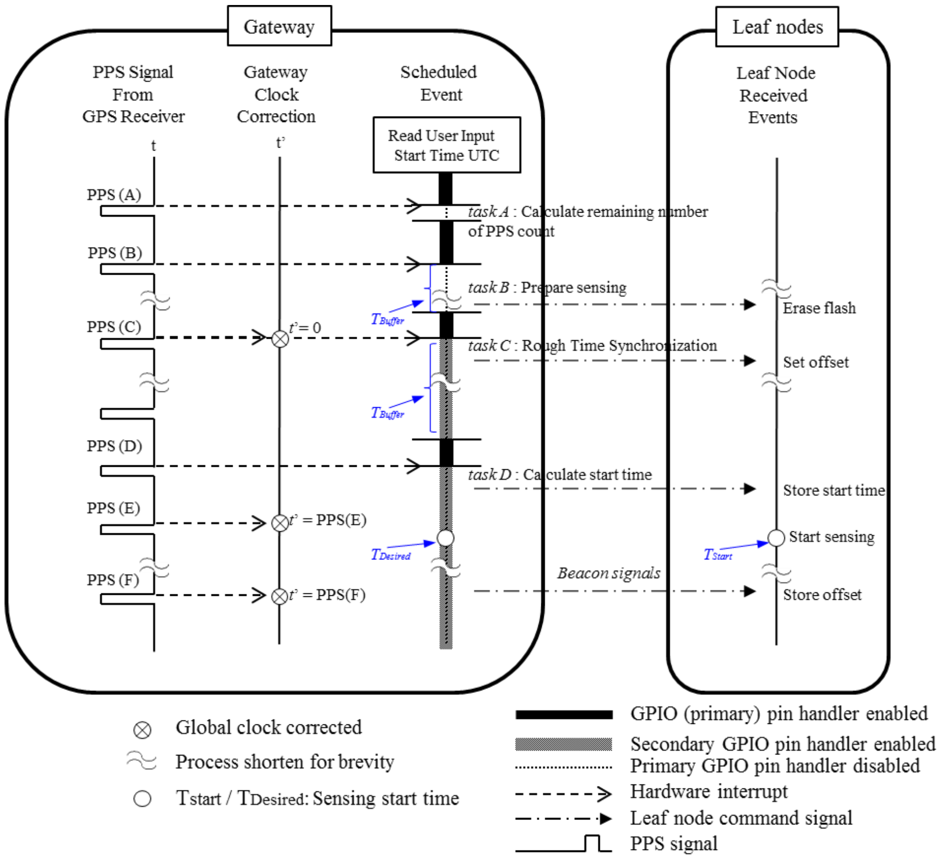

Gateway
Leaf nodes
PPS Signal From GPS Receiver
Gateway Clock Correction
Scheduled Event
Leaf Node Received Events
t
t'
Read User Input Start Time UTC
PPS (A)
PPS (B)
PPS (C)
PPS (D)
PPS (E)
PPS (F)
task A : Calculate remaining number of PPS count
task B : Prepare sensing
task C : Rough Time Synchronization
task D : Calculate start time
t' = 0
t' = PPS(E)
t' = PPS(F)
T_Buffer
T_Buffer
T_Desired
T_Start
Beacon signals
Erase flash
Set offset
Store start time
Start sensing
Store offset
Global clock corrected
Process shorten for brevity
Tstart / TDesired: Sensing start time
GPIO (primary) pin handler enabled
Secondary GPIO pin handler enabled
Primary GPIO pin handler disabled
Hardware interrupt
Leaf node command signal
PPS signal

Fig. 15 Flow of PPS_based _RemoteSensing algorithm

The flow of PPS_based_RemoteSensing is shown in Fig. 15, where the sequence of events occurring in a gateway (the left box), and in a leaf node (in the right box). The leftmost column in the gateway represents the PPS signal input coming from a GPS receiver to the gateway Imote2's GPIO pin. The middle column illustrates time flow in the gateway clock. The clock is adjusted to specific values as the algorithm requires. The rightmost column in gateway outlines the list of events scheduled at each step. Similarly, the column for the leaf nodes shows the list of events occurring upon receiving the commands from the gateway. The dashed arrows indicate the PPS signal interrupting the gateway hardware to trigger clock adjustment or initiate scheduled events. The dashed-dot arrows represent wireless communication from the gateway to leaf nodes to cast commands. The main function of this flow is to provide fundamental service used for obtaining synchronized measurements such as acceleration and strain.

To start measurements on participating leaf nodes in networks, the application requires users to input the same UTC time on all gateway nodes. Once the input parameters are entered, they are stored in the local flash memory. Then the primary GPIO interrupt handler is enabled and it waits for the PPS signal interrupts. To capture the first rising edge of the PPS signal, C_{END} (Fig. 12) is set to 1 at the beginning. The first PPS interrupt triggers the execution of *task A: Calculate* *remaining number of PPS count*[†] (see Fig. 15). This step ensures that the task is synchronized for all participating gateway nodes. Once the task A is successfully executed, the GPIO pin is disabled in order not to be further interrupted by consecutive PPS signals. At the same time, the gateway reads NMEA sentences received from the UART port. The sentence is parsed by every byte and the port will be closed when valid UTC information (T_{UTC}) is obtained. Finally, T_{UTC} is compared with the user defined start time ($T_{Desired}$). Time elapsed to perform those processes may vary on different Imote2 platforms on gateway nodes. Therefore, the next task is queued on following PPS interrupts so that the task can be executed at the same time for all participating gateway nodes. This restriction requires T_{Wait}, sufficient time before time reaches $T_{Desired}$; if the remaining time is larger than T_{Wait} i.e., $T_{Desired} - T_{UTC} > T_{Wait}$, the C_{END} is updated as $C_{END} = T_{Desired} - T_{UTC}$. Otherwise, if $T_{Desired} - T_{UTC} < T_{Wait}$ the user will be informed that $T_{Desired}$ should be larger. For example, T_{Wait} may be set to 5 seconds to let the gateway nodes finish the calculation.

After task A, the gateway nodes identify the participating nodes in the networks and prepare sensing through *task B: Prepare sensing*. The gateway first re-enables the primary GPIO interrupt hander and bypasses PPS signal interrupts till the counter reaches C_{END}, i.e., waits until $T_{Desired}$. At PPS (B) (Fig. 15) task B is executed and disables the GPIO pin not to be interrupted by following PPS signals. During the task B, two events are scheduled. 1) The gateway commands out to the participating leaf nodes to erase flash memories. 2) Set an oscillator-based timer on the gateway for T_{Buffer}. Although execution of the first event does not require precise task synchronization among multiple network, the latter requires synchronization. The purpose of T_{Buffer} is to allow leaf nodes successfully erase the already stored data before any sensing tasks so that measured data set can be stored. T_{Buffer} can be set for a non-integer value in second, such as 2.5 seconds. This buffer time will prevent gateways triggered by different PPS signals at the moment when T_{Buffer} expires, which may arise from processing time differences. Once the timer for T_{Buffer} is fired, the primary GPIO interrupt handler is re-enabled and C_{END} is set to 1 to capture the first rising edge of the PPS signal. A successful capture of C_{END} will disable the GPIO pin.

Periodic clock correction on the gateway and clock synchronization on leaf nodes are performed during *task C: Rough Time Synchronization*. When PPS (C) is reached, three events, which all requires task synchronization, are scheduled in the algorithm. 1) Global clocks of the gateway nodes are set to zero (*t*' = 0). This process ensures that all the gateway nodes set their internal clock to the same value. Because all sub-network processes the task at the same UTC time and the corresponding PPS interrupt, UTC information can be lost. 2) Start another timer T_{Buffer} and broadcasts a single beacon signal to the network containing global clock of gateway. A non-integer value in second for T_{Buffer} is required not to be interrupted with different PPS interrupt. Depends on the size of the network, T_{Buffer} can be set for 3.5 seconds for small sub-networks (less than 10 leaf nodes within visible range). A leaf node that receives the signal adjusts its offset so that network is roughly clock synchronized. Once tasks are executed, the primary GPIO interrupt handler is enabled again and waits for the next PPS signal interruption (C_{END} = 1). A successful capture of the PPS signal will disable the GPIO pin and move on to the following task, the task D. 3) Schedule a periodic counter (C_{Update}) to update global clock to prevent clock drift. This process timestamps the global time (T_{Prev}) and enables a secondary GPIO interrupt handler with a counter,

[†] An *Italic* font refers to the task identified in Fig. 15.

C_{Update}. For example, if C_{Update} is set to 5, the secondary GPIO handler bypasses four consecutive pulses and interrupts itself at the fifth PPS pulse. Then, global clock adjusts its clock by adding elapsed time to previously stored time information. Fig. 16 depicts the above process. The clock is assumed to be linear within C_{Update} and does not drift much. The process repeats till the sensing task is finished. The purpose of C_{Update} is to ensure synchronization among gateways by controlling clocks to increase according to PPS signals. Since the periodic updates of the gateway clocks are scheduled asynchronously as hardware interrupts, the clocks of the gateway nodes can be kept very accurate and hence accurately synchronized with each other.

Time for leaf nodes to actually start measurement will be commanded through *task D: Calculate start time*. When PPS (D) interrupts GPIO pin the gateway calculates T_{Start} and broadcast the time to the leaf node. When leaf nodes receive the message, waiting time (waitTime) is calculated based on leaf node current time (T_{Leaf}) and T_{Start} (i.e., $waitTime = T_{Start} - T_{Leaf}$). Since all leaf nodes in a sub-network are synchronized with the gateway clock at the beginning, leaf nodes start sensing roughly at the same time. Leaf nodes in multiple networks also start measurement nearly at the same time because gateway node clocks are initially synchronized and the Task D has been synchronized using the PPS signals.

The uncertainty of sensing start time in the networks will be compensated through resampling after sensing is finished. Gateway broadcasts beacon signals periodically and leaf nodes in the network stores offset and corresponding local time for regression analysis to estimate nonlinear clock drift when sensing is finished (Li, Nagayama *et al.* 2015). Finally, data synchronization, resampling, is applied after sensing is finished to eliminate the errors from uncertainties of start timing and randomness of the clock speeds.

5. Performance evaluation

Laboratory tests validated the proposed synchronized sensing protocols for both single network and multiple-network synchronized sensing. Tests are performed for short-term (1 min and 5 min) and long-term measurements (30 min) and measured random signals to compare the synchronization accuracy. Also, the stability of the clock is checked by compensating the clock drift with either linear or 5th order curve fitting. The results of the tests prove the prospective use of the algorithm for scalable WSS Networks aiming at long-term monitoring of civil structures.

```
Case (TaskC)
set.clock(0);   // Clock initialization
while( SensingTask == True) {  //Repeat if the sensing Task is not ended
     T_prev = get.Time();  // Read Time stamp
     // enable secondary GPIO pin
          if(count == Cupdate){
             time = T_prev+get.ClockRate()*Cupdate;
             // disable secondary GPIO pin
            }else
            count++;
}
```

Fig. 16 Periodic clock offset correction scheme

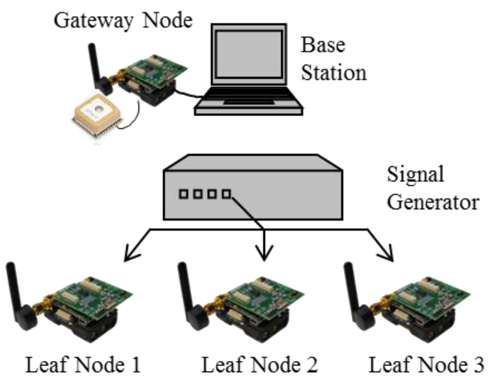


Fig. 17 Test setup for synchronized sensing accuracy estimation

5.1 Synchronized sensing protocols comparison in a single network

A small-scale network in which all leaf nodes can measure the identical random signals is prepared in an indoor laboratory to check the single network synchronized sensing (Fig. 17). The network consists of an Imote2 as a gateway and three Imote2s as leaf nodes. A GPS receiver is connected to the gateway and periodically updated the clock based on PPS signals (every 5 PPS signals) when proposed protocol was used. An ISM400 sensor board capable of measuring external analog signals is stacked on the Imote2 of each leaf node (MEMSIC, 2010). An analog signal generator, VibPilot (www.mpihome.com, 2010), generated Band Limited White Noise (BLWN) with 100 Hz bandwidth. To provide identical signals to three leaf nodes, the signal is split using connectors and fed to the sensor board. The test setup assures that all leaf nodes measure identical BLWN signals, which facilitate accurate estimation of synchronized sensing error in the measured data.

Synchronized sensing error can be quantified using the phase angle of the cross power spectrum density (CPSD) between the collected signals. Once the CPSD is calculated, the slope of the phase angle of the CPSD is found using linear curving fitting (Nagayama and Spencer 2007). Then, the synchronized sensing error is estimated by the following equation, in which θ is the angle of the phase angle curve.

$$\text{Time Synchronization error} = \frac{\theta}{2\pi} \times 10^6 \ (\mu s) \tag{1}$$

In calculation of CPSD, Hanning window with 1024 Fast Fourier Transformation points and 50% overlapping are used. The frequency range used for linear curve fitting the phase angle curve (the slope of phase angle) can be chosen based on the cut-off frequency used to measure the data with an ISM400 board. Throughout the experiment, sampling frequency of 100 Hz and corresponding cut-off frequency of 40 Hz is used. Thus, the frequency range used for linear fitting used is from 0 to 35 Hz. Fig. 18 shows an example of calculated phase angle of the CPSD between two measured signals. By linear curve fitting the phase angle the synchronized sensing with the frequency range 0 to 35 Hz is about 13 μs.

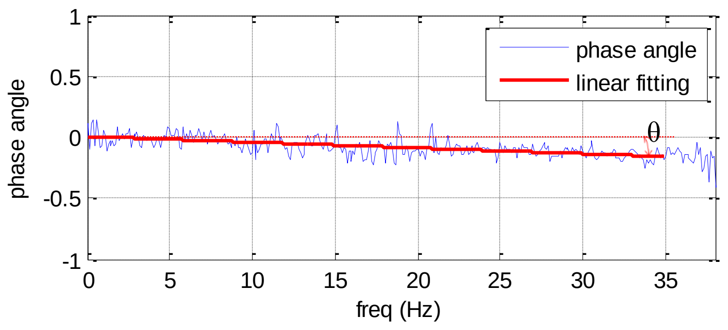


Fig. 18 Example of phase angle of the CPSD

Table 1 Synchronized sensing errors

Synchronized sensing protocol	Synchronized sensing error [μs] - Compared with base station local clock		
	1 min	3 min	30 min
Linear drift estimation	28.95	55.16	-1175.8
Nonlinear drift compensation (piecewise linear)	30.92	31.00	23.73
Nonlinear drift compensation (fifth order)	35.83	36.71	19.71
Proposed approach (linear)	35.26	35.28	34.05
Proposed approach (fifth order)	34.84	24.87	26.29

Three protocols are compared to validate the proposed synchronized sensing method in a single network: (i) synchronized sensing with linear drift estimation, (ii) synchronized sensing with nonlinear drift compensation, and (iii) the proposed algorithm. Both piecewise linear and fifth-order curve fitting were used for the comparison in the nonlinear drift compensation algorithm and proposed protocol. The sampling frequency of data acquisition with WSSs was 100 Hz, and sensing durations were varied from short-term (1 min), mid-term (3 min), and to long-term (30 min) measurements. As shown in Table 1, when the error was calculated based on the base station local clock, similar degree of synchronized sensing error, about 33 μs on average, are observed in all protocols for short-term tests. The Linear drift estimation showed increased synchronized sensing error (over 50 μs) for mid-term test, whereas other methods showed similar error sizes with short-term tests. Significant effects of nonlinear clock drifts in the leaf nodes are observed in long-term tests: Two sensor measurements are unsynchronized showing the synchronization error as large as 1 ms. Nonlinear clock drift compensation ensured that the similar degrees of synchronized errors in long-term measurements for other cases. The higher-order regression of the clock drift enhanced the performance; an approximately 15% increase for both protocols. The results validated that the proposed method provides synchronized sensing with error less than 50 μs in a single network for an arbitrary length of time.

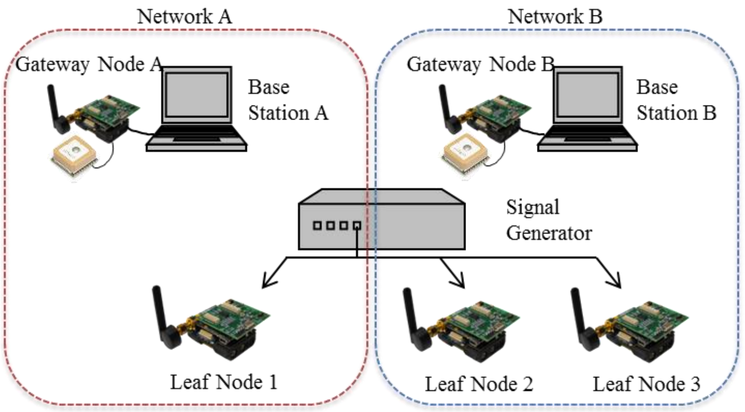


Fig. 19 Test setup for evaluating synchronized sensing for multiple networks

5.2 Multiple-network synchronized sensing accuracy

Two WSS networks, as shown in the Fig. 19, are prepared to validate two main goals of the proposed algorithm: (i) periodic correction of gateway node's clocks with PPS signals to compensate nonlinear drift, and (ii) achieve synchronized sensing between different sub-networks. In the test setup, Network A consists of one leaf node, Leaf Node 1, and Network B consists of two leaf nodes, Leaf Node 2 and Leaf Node 3. Each gateway node is connected to a base station computer and a GPS receiver. Each leaf node's Imote2 is connected to an ISM400 sensor board to measure external analog signals. Since the two networks are closely located, two different radio communication channels are designated to each network. When the frequency range of the selected radio channels are apart from each other, mutual interference can be avoided (Linderman, Rice *et al.* 2010). Synchronization accuracy between the two networks is estimated by comparing the signals measured by the sensor boards. To provide the same signal to all sensors, random signals generated from VibPilot are split and fed into the 4th channel of the ISM400 boards. The sampling frequency of all wireless sensors is 100 Hz.

Table 2 Multiple-network synchronization error comparison

Synchronized sensing protocol		Multiple-network synchronized sensing error [μs]		
		1 min	3 min	30 min
Linear curve fitting	Average	19.55	21.06	33.49
	<Std.[*]>	<4.23>	<6.25>	<5.09>
Fifth-order curve fitting	Average	16.21	11.08	30.60
	<Std.[*]>	<4.40>	<6.45>	<2.43>

[*] Std. refers to the standard deviation of synchronization error.

The first test demonstrated the effectiveness of periodic correction of gateway node’s clocks by comparing the measurements without and with gateway clock adjustments. At the beginning of sensing, both gateway nodes are clock synchronized to the UTC time. However, the differences between the clock frequencies of the two gateway nodes are not corrected and are broadcasted to their leaf nodes. The phase angle of the CPSD calculated using signals measured for 3 min period showed relatively large error of −138 μs based on the phase angle with the frequency range between 0 to 35 Hz. When both gateway clocks are updated periodically after every 5 PPS signals during the sensing period and are broadcasted to leaf nodes, the slope of the phase angle of the CPSD is much smaller. The synchronization error was −20 μs based on the phase angle with the frequency range. The results indicate that the clock drift on the gateway nodes is not negligible and their clocks should be adjusted to provide synchronized sensing among multiple networks.

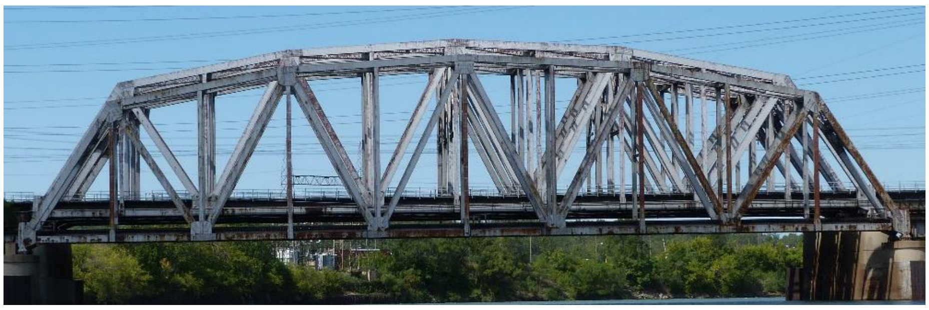
Further tests are performed for extended sensing periods to validate synchronized sensing between multiple networks. The test combinations have different sensing durations (1 min, 3min, and 30 min) and different clock drift compensation (linear and 5th order curve fitting). Table 2 summarizes averages and standard deviations of the synchronization errors over a number of tests for each test combination. Accurate synchronized sensing, which is less than 50 μs, is obtained from the leaf nodes in different sub-networks. Moreover, the error size did not increase significantly as the sensing time increases indicating the stability of the protocol. Due to periodic gateway node clock correction, which assured the linearity of the clock drift, linear curve fitting result showed similar degree of accuracy compared with 5th order curve fitting. In summary, both single network and multiple-network test results achieved synchronized sensing with less than 50 μs errors. Therefore, the proposed protocol is suitable for long-term monitoring of scalable WSS networks.

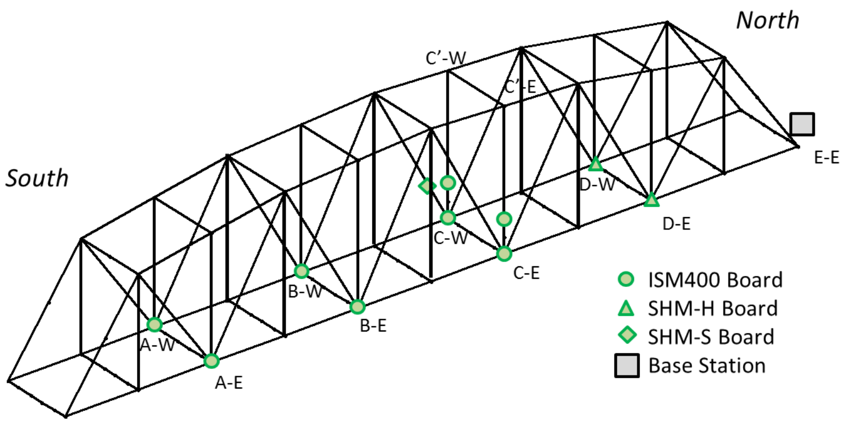
5.3 Full-scale implementation

This section presents an example of application, in which the developed time synchronization protocol is applied to a railroad bridge. Because of the repetitive nature of train wheel loads, deterioration of railroad bridges due to vehicle-bridge interaction has been of concern (Yang, Yau *et al*. 2004, Kim, Lynch *et al*. 2011). Numerous analytical solutions have been made, while obtaining data from the bridge which are synchronized with those from a train in field tests is intrinsically challenging. Thus, field tests were performed, which can measure both response of the bridge and the input. Such a field test was impossible until the proposed synchronized sensing protocol is developed. Here two geographically separate sensor networks are prepared: One on a railroad bridge and the other on a train for monitoring responses of a railroad bridge and input loading. The first network was a full-scale network deployed on a selected railroad bridge, shown in Fig. 20. Six WSS nodes, each with an ISM400 board having a measurable range up to ± 2 g, were deployed at each nodal points (A-E and -W, B-E and -W, and C-E and -W in Fig. 20) of the truss bridge to provide global behavior. Another network was prepared on the locomotive of a train, which was composed of following seven freight cars. A laptop inside the engine reached the gateway node on the side rail and controlled the train sensors. Three leaf nodes shown in Fig. 21 captured the responses of the front and rear bogies, as well as the responses of the car body.

An example of measured signals while the test train crossed the bridge is shown in Fig. 22.

Leaf nodes in two networks initiated sensing with the same input; same UTC time, sampling frequency (25 Hz) and measurement period (over 10 min). The test train reached the bridge at 5 MPH from the South. When the train fully crossed the bridge, it fully stopped. After a minute of rest, the train re-approached the bridge and stopped in the middle of the bridge for about 20 seconds. The train exited the bridge by moving towards the South end of the bridge. As can be seen in Fig. 22, acceleration signature of the bridge agrees well with the train log indicating that two networks were synchronized with each other. These results demonstrate the potential of using synchronized data to understand bridge responses considering the interaction dynamics with the freight train. For example, Table 3 summarizes identified first five frequencies of the bridge records under ambient vibration and with a test train sitting on the bridge. At the same time, the PSD of the vehicle in stationary condition give clear periodic peaks in every 4.9 Hz, implying that the frequency is related to the engine vibration (Fig. 23(a)). On the other hand, when the train ran over the bridge at 40 km/h, no significant peak was observed throughout the measured region (Fig. 23(b)). Thus, along with the weight of the train, its vibration can be seen as the possible sources for the changes of the identified bridge frequencies. Thus, in the future, the proposed algorithm can be used for understanding of the integrity of the bridge by, for instance, extracting the forced responses.


(a) Bridge network



(b) Deployment map of the bridge network

Fig. 20 Bridge network

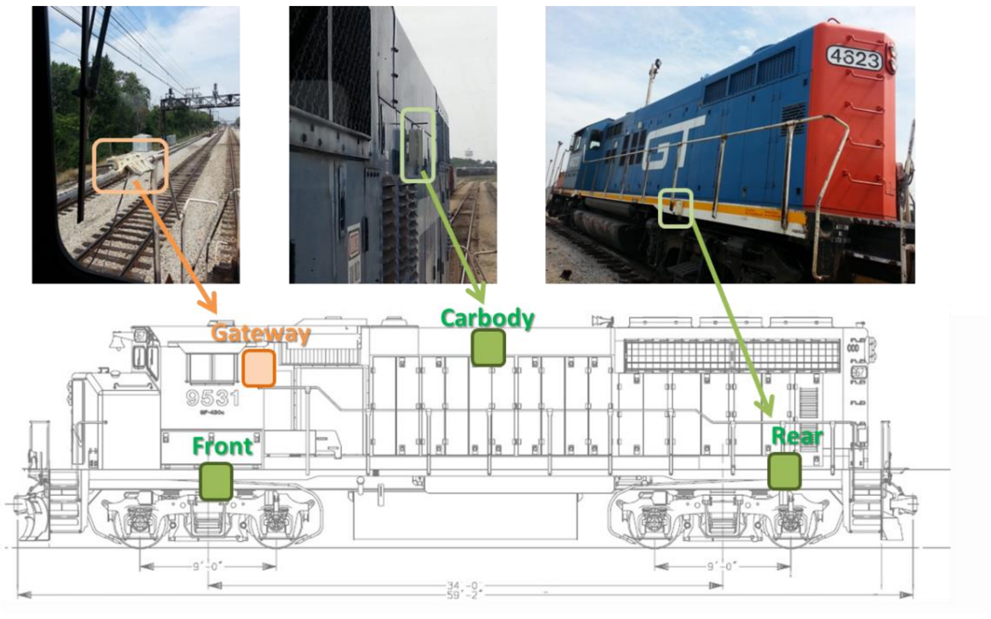


Fig. 21 Train sensors setup

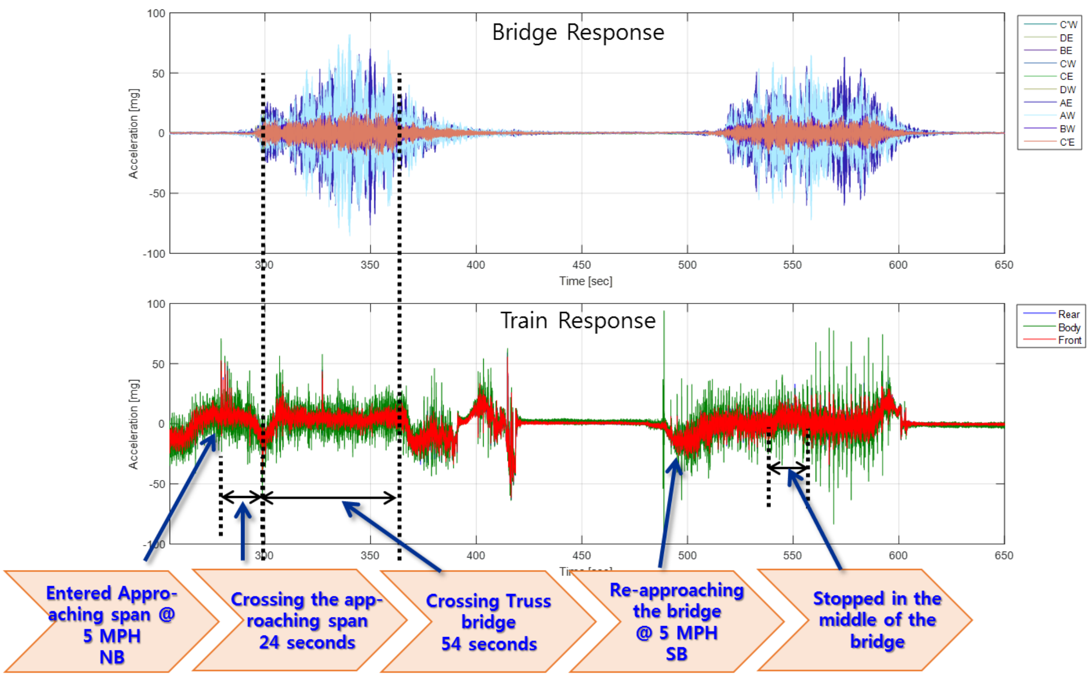
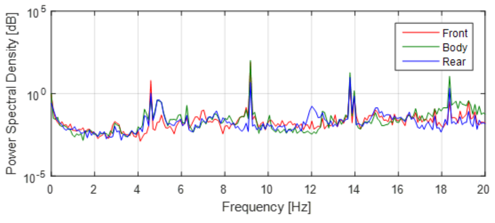
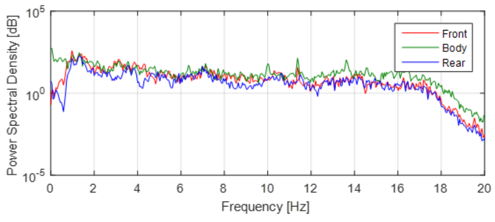


Fig. 22 An example of synchronized sensing



(a)



(b)

Fig. 23 The power spectral density of vertical vibration of a (a) train sitting and (b) train running

Table 3 First five peaks in bridge auto power spectral density

Ambient vibration [Hz]	Train sitting on the mid-span of the bridge [Hz]
1.56	1.27
3.0	1.8
3.42	2.8
5.27	3.26
6.3	4.59

6. Conclusions

Advances in wireless sensing technologies have attracted great attention for structural health monitoring applications. Successful use of wireless smart sensors in such applications requires high-accuracy and long-term synchronized sensing to ensure the quality of the data. Traditional approaches often provide only clock synchronization among WSSs, which does not necessarily yield synchronized data. Although the internal clocks in WSSs are precisely synchronized, random errors in the start of sampling and clock rates can introduce subsequent synchronization errors. Synchronized sensing protocols have been developed, but each has certain limitations. In this paper, a scalable protocol that provides effective synchronized sensing among multiple networks has been presented. The protocol used the Pulse-Per-Second signals from low-cost GPS receivers. The method was implemented on the Imote2 sensor platform and achieved two following main goals; i) periodic clock correction of gateway nodes to compensate nonlinear clock drift, and ii) achieve 50 μs synchronized sensing accuracy from leaf nodes in different networks. The results demonstrated that the proposed approach is highly scalable, realizing precise synchronized sensing for an arbitrary duration of data collection.

Acknowledgments

The authors gratefully acknowledge the support of this research by the National Science Foundation under Grant No. OISE-1107526 and the partial support by National Science

Foundation under Grant CMMI-0724172 (NEESR-SD) and the Federal Railroad Administration under the BAA 2010-1 project.

References

Chen, Y., Wang, Q., Chang, M. and Terzis, A. (2011), "Ultra-low power time synchronization using passive radio receivers", Information Processing in Sensor Networks (IPSN),Chicago, IL, USA.

Cho, S.J., Jo, H., Jang, S., Park, J.W., Jung, H.J., Yun, C.B., Spencer, Jr. B.F. and Seo, J.W. (2010), "Structural health monitoring of a cable-stayed bridge using wireless smart sensor technology: data analyses", *Smart Struct. Syst*., **6**(5-6), 461-480.

Elson, J. and Römer, K. (2003), "Wireless sensor networks: A new regime for time synchronization", *Comput. Commun. Rev*., **33**(1), 149-154.

Elson, J., Lewis, G. and Deborah, E. (2002), "Fine-grained network time synchronization using reference broadcasts", *ACM SIGOPS Operating Systems Review*, **36**(SI), 147-163.

Gay, D., Levis, P., Culler, P.D. and Brewer, E. (2003), "nesC 1.1 Language Reference Manual", http://today.cs.berkeley.edu/tos/tinyos-1.x/doc/nesc/ref.pdf .

GlobalTop (2010), www.gtop-tech.com

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 Struct. Syst*., **6**(5-6), 439-459.

Jo, H., Sim, S.H., Mechitov, K.A., Kim, R.E., Li, J., Moinzadeh, P., Spencer, Jr. B.F., Park, J.W., Cho, S., Jung, H.J., Yun, C.B., Rice, J.A. and Nagayama, T. (2011), "Hybrid wireless smart sensor network for full-scale structural health monitoring of a cable-stayed bridge", *Proceedings of SPIE*, San Diego CA, USA, March.

Kim, J., Lynch, J.P., Lee, J.J. and Lee, C.G. (2011), "Truck-based mobile wireless sensor networks for the experimental observation of vehicle–bridge interaction", *Smart Mater. Struct*., **20**(6), 065009.

Kim, R.E., Nagayama, T., Jo, H. and Spencer, Jr. B.F. (2012), "Preliminary study of low-cost GPS receivers for time synchronization of wireless sensor networks", *Proceedings of SPIE*, San Diego CA, USA, March.

Li, J., Nagayama, T., Mechitov, K.A. and Spencer, Jr. B.F. (2015), "Efficient time synchronization for structural health monitoring using wireless smart sensor networks", *Struct Control Health Monit.*, DOI: 10.1002/stc.1782

Linderman, L.E., Rice, J.A., Barot, S., Spencer, B.F. and Bernhard, J.T. (2010), "Characterization of wireless smart sensor performance", NSEL report, NSEL Report Series Report No. 021.

Lynch, J.P. (2007), "An overview of wireless structural health monitoring for civil structures", *Philos. T. R. Soc. A.*, **365**(1851), 345-372.

Lynch, J.P. and Loh, K.J. (2006), "A summary review of wireless sensors and sensor networks for structural health monitoring", *Shock Vib Digest*, **38**(2), 91-130.

Lynch, J.P., Law, K.H., Kiremidjian, A.S., Carryer, E., Farrar, C.R., Sohn, H. and Wait, J.R. (2004), "Design and performance validation of a wireless sensing unit for structural monitoring applications", *Struct. Eng. Mech*., **17**(3-4), 393-408.

Lynch, J.P., Wang, Y., Law, K.H., Yi, J.H., Lee, C.G. and Yun, C.B. (2005), "Validation of a large-scale wireless structural monitoring system on the Geumdang bridge", *Proceedings of the Int. Conference on Safety and Structural Reliability*, Rome, Italy.

Lynch, J.P., Wang, Y., Loh, K.J., Yi, J.H. and Yun, C.B. (2006), "Performance monitoring of the Geumdang Bridge using a dense network of high-resolution wireless sensors", *Smart Mater. Struct.*, **15**(6), 1561.

Mannermaa, J. (1999), "Timing performance of various GPS receivers", *Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and the IEEE International Frequency Control Symposium*.

Maróti, M., Kusy, B., Simon, G. and Lédeczi, Á. (2004), "The flooding time synchronization protocol", *Proceedings of the 2nd international conference on Embedded networked sensor systems*, Baltimore, MD, USA, November.

Mills, D.L. (1991), "Internet time synchronization: the network time protocol", *IEEE T. Commun.*, **39**(10), 1482-1493.

Nagayama, T. and Spencer, Jr. B.F. (2007), "Structural Health Monitoring Using Smart Sensors", Newmark Structural Engineering Laboratory Report Series, No. 001.

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

Nagayama, T., Spencer, Jr. B.F. and Fujino, Y. (2008), "Synchronized sensing for structural health monitoring using smart sensors", *Proceedings of the SMSST'07 World Forum on Smart Materials and Smart Structures Technology*, China, May.

Pakzad, S.N., Fenves, G.L., Kim, S. and Culler, D.E. (2008), "Design and implementation of scalable wireless sensor network for structural monitoring", *J. Infrastruct. Syst.*, **14**, 89-101.

Rice, J.A. and Spencer, Jr. B.F. (2009), "Flexible Smart Sensor Framework for Autonomous Full-scale Structural Health Monitoring", NSEL report, NSEL Report Series Report No. NSEL-018.

Roche, M. (2006), "Time Synchronization in Wireless Networks", CSE574S: Advanced Topics in Networking: Wireless and Mobile Networking, Washington University in St. Louis.

Sazonov, E., Krishnamurthy, V. and Schilling, R. (2010), "Wireless intelligent sensor and actuator network-a scalable platform for time-synchronous applications of structural health monitoring", *Struct Health Monit.*, **9**(5) 465-476.

Spencer, Jr. B.F., Nagayama, T. and Rice, J.A. (2008), "Decentralized structural health monitoring using smart sensors", *Proceedings of SPIE*, San Diego CA., USA, March.

VibPilot (2010), m+p international, www.mpihome.com

Wang, Y., Lynch, J.P. and Law, K.H. (2005)m "Wireless structural sensors using reliable communication protocols for data acquisition and interrogation", Ann Arbor, 1001:48109.

Windl, U., Dalton, D., Packard, H. and Martinec, M., "The NTP FAQ and HOWTO Understanding and using the Network Time Protocol", http://www.ntp.org.

Yang, Y.B., Yau, J.D. and Wu, Y.S. (2004), "Vehicle-bridge interaction dynamics", World Scientific Publishing Company.

CY