Energy Efficient Time Synchronization in WSN for Critical Infrastructure Monitoring

Aravinda S. Rao[1], Jayavardhana Gubbi[1], Tuan Ngo[2], James Nguyen[2], and Marimuthu Palaniswami[1]

[1] ISSNIP, Dept of Electrical and Electronic Engineering, The University of Melbourne, Vic - 3010, Australia
{sridhara, jgl, palani}@unimelb.edu.au
[2] Infrastructure Protection Research Group, Dept of Civil and Environmental Engineering, The University of Melbourne, Vic - 3010, Australia
dtngo@unimelb.edu.au, james.idt@gmail.com

Abstract. Wireless Sensor Networks (WSN) based Structural Health Monitoring (SHM) is becoming popular in analyzing the life of critical infrastructure such as bridges on a continuous basis. For most of the applications, data aggregation requires high sampling rate. A need for accurate time synchronization in the order of $0.6-9$ μs every few minutes is necessary for data collection and analysis. Two-stage energy-efficient time synchronization is proposed in this paper. Firstly, the network is divided into clusters and a head node is elected using Low-Energy Adaptive Clustering Hierarchy based algorithm. Later, multiple packets of different lengths are used to estimate the delay between the elected head and the entire network hierarchically at different levels. Algorithmic scheme limits error to 3-hop worst case synchronization error. Unlike earlier energy-efficient time synchronization schemes, the achieved results increase the lifetime of the network.

Keywords: Time Synchronization, Wireless Sensor Networks, Critical Infrastructure Monitoring, Structural Health Monitoring, Energy Efficient.

1 Introduction

Time synchronization in WSN has been an important research area over the past decade. Numerous protocols, benchmarked algorithms and approaches have been proposed to reduce the synchronization error and also being implemented to test their viaibility for WSN; Reference Broadcast Synchronization (RBS) [2], Timing-sync Protocol for Sensor Networks (TPSN) [3], Flooding Time Synchronization Protocol (FTSP) [10], Lightweight Time Synchronization for Sensor Networks (LTS) [4], Tiny-sync [15] and Mini-sync [15] are widely used.

Time synchronization is an inherent problem in any network. The time between any two networked computers differs due to clock drift and clock offset. Often the crystal oscillator, provider of appropriate timing signals, is observed to be drifted from its normal specified frequency. The clock instability is caused

by many environmental factors, chiefly attributing to temperature variation [13]. The frequency offset is the difference in clock timing between two clocks at any particular instant. As the frequency of the crystal varies, the clock offset varies i.e. offset increases or decreases relative to other clock.

Physical time plays a crucial role in WSN application for Critical Infrastructure Monitoring (CIM) [1, 8]. Multiple nodes possessing identical timestamps of an event are essential. The significance of physical time with respect to WSNs are detailed in [12] as : (a) interaction between a sensor network and the observer, (b) interaction between nodes, and (c) interface between nodes and the real world. Absence of real-time clock on WSN motes, like Network Time Protocol (NTP) [11] on the Internet, having accuracy in the order of nano to pico seconds, has given rise to software schemes to synchronize their on-board clocks. Numerous solutions exist in literature such as software clocks, unidirectional synchronization, round trip synchronization and reference broadcasting having many advantages and disadvantages; schemes based on high-rate data collection are shown to have decreased in errors to failure in synchronizing [12]. Power-constrained frequent synchronization is required to tackle drifting, a major contributor to error, with an accuracy of 0.6 to 9 μs for modal analysis [9]. A 5 ms drift in 6 s period was also reported by Wang *et al.* [17].

In this paper, an energy-efficient time synchronization algorithm is proposed using few existing algorithms, but combines them uniquely to achieve the desired goal. At first, it uses Low-Energy Adaptive Clustering Hierarchy (LEACH) for dividing the network into clusters based on nodes' available energy [5]. Later, a new algorithm is proposed for synchronization. This paper is organized as follows: Section 2 gives brief description of existing protocols, Section 3 contains the proposed approach towards time synchronization followed by Section 4 with results and discussion, and the work is concluded in Section 5.

2 Related Work

Although, in general, substantial amount of work has been carried out toward time synchronization [16], however, it was during the nascent stages of WSN research, consequently, less importance was paid to energy consumption. As mentioned by Krishnamurthy *et al.* in [9], there is a need for frequent synchronization than anticipated, affecting network lifetime. Following paragraphs presents few existing useful time synchronization methods.

Reference Broadcast Synchronization (RBS) is a receiver-to-receiver synchronization, a pioneering work, has a main drawback of growing number of exchanged messages with larger networks [2]. With Flooding Time Synchronization Protocol (FTSP), flooding the network with synchronization packets to the neighbors, fails to compensate the propagation delay occurred and requires a node to have enough data to perform linear regression to synchronize [10]. Timing-sync Protocol for Sensor Networks (TPSN) [3] uses two phases: Level discovery phase forms a hierarchical network of different levels. Node with level 0 forms the top level followed by level 1, 2, 3 and so on. Node at level i commu-

nicates with at least a node at the level above (*i-1*) it. During synchronization phase, the node at level 0 (root node) initiates the phase using constrained flooding [16, 3]. In TPSN, though a node at non-immediate level overhears the message from higher levels, it only synchronizes with a node at a level above it.

Lightweight Time Synchronization for Sensor Networks (LTS) uses three messages to synchronize a pair of nodes [4] during the pairwise synchronization stage. It is expandable from single-hop to multi-hop network, from centralized to distributed synchronization. The tree-depth dependency and assumption of an equally bounded-clock for all nodes limits appropriate synchronization. In the recent past, Shahzad *et al.* [14] proposed Energy Efficient Time Synchronization (EETS) combining RBS and TPSN for achieving better energy efficiency. Although EETS improves energy efficiency, particularly in large networks with multi-hops reaching up to 8-10 levels, the synchronization error crosses higher than required level for CIM. More recently, Kim *et al.* [7] have developed an energy conserving algorithm by reducing the number of packets; another method combining RBS, TPSN and LTS methods, the topology is identical to TPSN and the best achievable error rate is affected by number of hops. In our proposed algorithm, energy efficient clusters are created using LEACH as a part of first stage. The second stage combines the aforementioned class of algorithms to form hierarchical levels as in TPSN and pairwise synchronization as in LTS to synchronize time.

3 Approach

Energy-efficient time synchronization is carried out in two stages: (a) dividing the network into clusters in an energy-efficient manner using LEACH [5] and (b) synchronizing the time within the cluster using multi-level pairwise synchronization. LEACH elects a cluster head periodically based on the highest energy-available live node. The chance of becoming a cluster head is rotated randomly ensuring that energy from a single node is not drained out [5]. Upon cluster head election, pairwise transmission-range based multilevel, hierarchical sub-clusters are formed in the vicinity of the parent node. Variable-length packets are used to determine the nondeterministic latency between sender and the receiver and compensate this delay before transmission. The algorithm ensures that the time of a child node is same as that of the its parent. The elected four cluster heads and the cluster formation is pictorially represented in Figure 1.

According to LEACH algorithm [5], the optimal number of nodes to be cluster heads was chosen to be $\hat{N} = 5\%$. In case of the proposed algorithm, as the number of nodes in any cluster is limited to 12 (empirically chosen), $\hat{N} = (\text{Total Number of Nodes})/12 + 1$ is chosen for CIM. The second part of time synchronizing procedure is divided into four phases as explained in the following subsections.

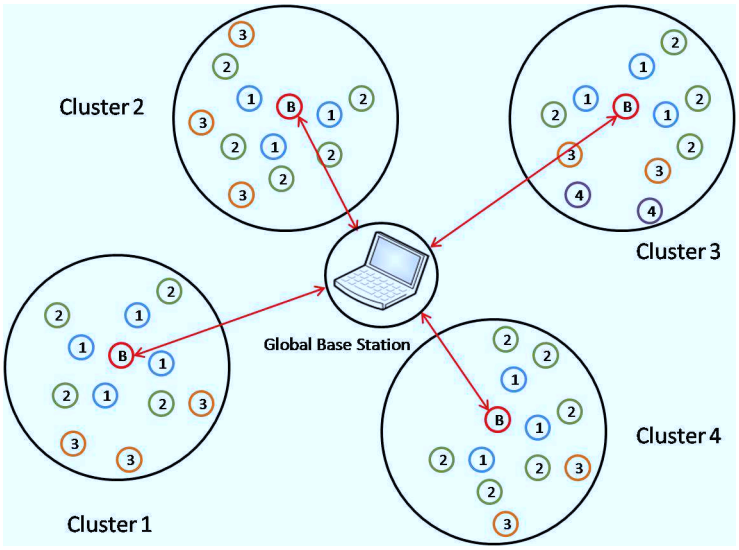

Fig. 1. Schematic of Cluster formation using LEACH. Numbers within the node represent levels. B is the chosen cluster-head in any given round.

3.1 Initiation

Let there be 'N' nodes neighboring to base station node. Out of 'N' nodes, only 'P' of them are within the communication range of base station node and the remaining (N-P) nodes are not. All nodes in the network are switched ON before any communication or network activity is initiated. Node B initiates its routine by synchronizing its time with the base station i.e. with the computer ensuring database connection. It should be noted that the nodes in the topology are placed such that it is in the communication range of at least one node.

3.2 Registration

The process of adding a new node to the topology is called as 'Registration'. Each node has its own ID and are uniquely identifiable. For discussion, let us consider $N = 2$. Nodes in the transmission range of B are therefore 1 and 2 and vice-versa. Therefore, the transmission power for both parent (B) and child nodes (1 and 2) are the same. After initiation process, B broadcasts a READY signal to inform its neighbors that it is ready to accept REQUEST from child nodes. Now, upon hearing B's broadcast signal, both 1 and 2 will send a REQUEST signals. When a READY signal is sent, B will wait for t_w seconds to hear from the nodes. Any message received after t_w is discarded. Suppose, B received both the requests from 1 and 2, then B will serve first node 1 folowed by 2 in a Time Division Multiple Access (TDMA) slotted fashion. Nodes 1 and 2 are waiting for the COMMAND signal from B. Nodes 1 and 2 will not proceed further until they receive any signal from B.

3.3 Calculating Nondeterministic Latency

Upon receiving and processing the COMMAND signal from B, node 1 will wait for the t_{start} time to start. At t_{start} time, it will send a message $M1$ to B. It is repeated for n times at every t_{repeat} interval. Packets are time stamped at physical layer of node 1 and are stored in B as and when received. Next, B will send a COMMAND signal with $M2 > M1$ greater i.e. the message length greater than the previous one. Even this time, B records node 1's data. Let δt_n be the time difference between consecutive packets arrived at the base station from a particular node. Let L_{M1} be the message length of $M1$ and the time stamp of the packets' arrival be $t_1 ... t_n$. Time difference between packets can be calculated as in (1), where n is the packet arrived at time t_n. B calculates the average time d_{M1} required by a packet of size $M1$ to reach base station due to send time, access time, propagation time and receive time and is given by (2). Let L_{M2} be the length of M2. Now, B sends a second COMMAND signal with $M2 > M1$. Equations (1) and (3) are calculated for $M2$. Then essentially, the time required for packet of size $L_{M2} - L_{M1}$ to be constructed, transferred and received from node 1 (or node 2) to B is given by (4). Now B sends t_{Byte} in a REGISTRATION_COMPLETE message to node 1. t_{Byte} is used by a node to compensate the delay for M bytes of data whenever in future it wishes to send the data to B. B accept packets strictly from level 1, and level nodes 1 accept data from level 2 only, identical to TPSN [3]. After this, node 1 sends a READY command to its neighbors. At the same time, B sends a COMMAND signal to node 2 and the process continues. At the end of REGISTRATION_COMPLETE with 2, B sends an ACCEPT command to accept data from 1 and 2. Node 2 continues to act as parent and starts broadcasting READY signal to its neighbors.

$$\delta t_n = |t_n - t_{n-1}| \tag{1}$$

$$d_{M1} = \frac{\sum_{i=1}^{n} \delta t_n}{n} \tag{2}$$

$$d_{M2} = \frac{\sum_{i=1}^{n} \delta t_n}{n} \tag{3}$$

$$t_{byte} = \frac{(d_{M2} - d_{M1})}{(L_{M2} - L_{M1})} \tag{4}$$

3.4 Time Synchronization

After every T seconds predetermined by application and the frequency of data collection, B multi-casts current time to all the registered nodes. T is calculated using (5), where t_i is the time required for a i^{th} node to be registered, t_{accept} is the time for accepting data from all the registered nodes. The typical tree structure formed during time synchronization after selecting the cluster head is illustrated in Figure 2.

$$T = \frac{\sum_{i=1}^{N} t_i}{N} + t_{accept} \tag{5}$$

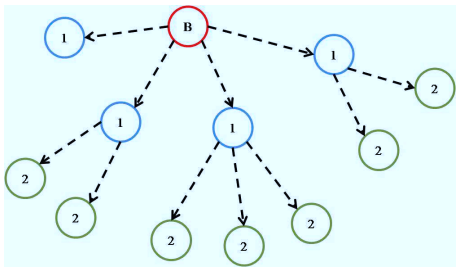

Fig. 2. Hierarchical level creation during proposed second stage of time synchronization

The term $\frac{\sum_{i=1}^{N} t_i}{N}$ is required if a node child node fails and wants to re-register. In (5), first part is 'Registration time' and the second part is 'Data acceptance time' in the proposed scheme. Additionally, if there is a new node, then it can register itself in the first part and send data subsequently. In general, T should be selected such that allowances are made for any new node to be added.

3.5 Packets required to Synchronize a Node

Determining number of packets 'n' in the COMMAND is a two-step process. During the first step, B sends COMMAND with n=5. The maximum tolerable synchronization error between two nodes is specified by the user as t_{error}. During the second step, the node calculates the $\delta Error_n$ and the average of $\delta Error_n$ as in (6) and (7) respectively. If $\delta Error_{avg}$ is more than predetermined value δt_{error}, then B raises its bar and requests more packets during M2 acquisition, otherwise M2 will have 'n' packets as well; B tries to find the delay due to variable-length packet size.

$$\delta Error_n = \delta t_n - t_{repeat} \tag{6}$$

$$\delta Error_{avg} = \frac{\sum_{i=1}^{n} \delta Error_n}{n} \tag{7}$$

4 Results and Discussion

The proposed energy-efficient time synchronization algorithm can be categorized as peer-to-peer, externally synchronized, deterministic, sender-to-sender and clock-corrected approach analogous with table 2 of [16]. It gives an added advantage of selectively using the node head for energy efficiency. Combining well-known schemes effectively, the result presented is based on theory and from earlier papers. Having presented the simulation results in this paper, in future, we plan to validate the proposed algorithm on iMote2 platform and the implementation is underway. The reasons for choosing iMote2 platform, apart from being available off-the-shelf, due to large memory capacity and high processing power of iMote2s. In SHM application (such as bridges), continuous collection of 3-axis

accelerometer data is required usually at 100-200 Hz. Furthermore, processing and storage play vital roles. Hence, to channelize the data to a base station, an energy-efficient data aggregation procedure is schemed to perform periodically. For CIM, the distance between nodes is often limited to 50 m: iMote2's antenna range (external) is 60 m, an ideal platform for CIM applications. At the end of this section, theoretical energy consumption and effect of packet length with respect to iMote2 platform are presented.

LEACH is a very popular clustering-based routing protocol, has been shown to minimize global energy usage by distributing the load among nodes [5]. Theoretically and experimentally, it has been shown that the lifetime of the network is increased 3 times compared to a static topology. In the first stage of the proposed time synchronization, LEACH based clustering is used for dividing the network into clusters. From the results presented in [2, 4, 3, 10], it is very clear that the clock error increases as the number of hops increase. Using cluster-based step at the first stage, ensures that the number of hops required in the second stage of the algorithm will never cross three levels (12 nodes). Hence, the maximum error in time synchronization by the proposed algorithm is the worst case 3-hop error that of TPSN and LTS. According to Krishnamurthy *et al.* [9], anything less than 10 μs error for 30 minutes duration is acceptable for modal analysis using acceleration sensors. By virtue, the proposed algorithm ensures this criterion. Moreover, the clustering algorithm enables data aggregation using energy-efficient routing apart from time synchronization.

4.1 Simulation Analysis

The experiment was conducted using OMNeT++, a component-based discrete event network simulation package. OMNeT++ offer tools to simulate computer networks, queuing networks, processor architectures and for distributed systems. We used NICTA's Castalia 3.1 framework on top of OMNeT++ 4.1 to simulate WSN scenarios [6]. In particular, for this work, we developed three simulation scenarios with three nodes - one base base station node (N0) and two sink nodes (N1, N2). N0 is the head of the nodes and the time-provider for other two nodes. The experiment was simulated for 10 s, with synchronization period of 1 s each for a total period of 10 s. Start up delay for the nodes were initialized to 0.5 ms. Node N0 waited for N1 and N2 for their reply after sending the initiation (READY) signal. The waiting time (t_w) was kept at 400 ms following the broadcast of READY signal to allow sufficient time to respond. Both N1 and N2 nodes replied with REQUEST signal. After t_W seconds, N0 processed REQUEST in ascending order of the node IDs. Node N0 sent a COMMAND signal to N1 with t_{start}=10 ms, t_{repeat}=10 ms, n=5, M1=5, M2=15. N1 sent messages to N0 after adding current time with t_{start} at t_{repeat} intervals (M1=5, M2=15 bytes of data sent 'n' times).

Node N0 calculated δt_n for M1 and M2 separately and then the average of δt_n i.e. $dM1$ and $dM2$. t_{Byte} was calculated using equation (4). Node N0 calculated t_{Byte} and $\delta Error_{avg}$, and sent the t_{Byte} to N1 as variable-length packet delay with REGISTRATION_COMPLETE signal. After processing nodes N1 by N0,

N2 was processed by sending COMMAND signal. Further, after processing N2, node N0 sent ACCEPT signal at start of t_{accept}. In our simulation t_{accept}=0 s as only two sink nodes were considered. At T = 1 s from the start of the READY signal, N0 multi-casts current time to registered nodes. Following this, N0 sent READY signal for the next round of clock synchronization and this process of synchronization was continued at different levels. We simulated for only one level i.e. between N0 and N1-N2. In this simulation for each time synchronization cycle, nodes N0 and N1 sent REQUESTs and participated in registration; however, during implementation, registration will be done once only. Media Access Control (MAC) layer was bypassed in the simulation environment; radio was in Ideal mode with a transmission power level of 0 dBm and configured to zero interference model; wireless channel was set to have bidirectionally identical signal quality links between nodes. One-hop routing was carried out by application layer.

- **Scenario 1 - Equidistant nodes:** N1 and N2 with distance equal to 14.14 m from N0 and t_{repeat} = 10 ms; for Node N1 t_{Byte} was 8.27 ms, $\delta Error_{avg}$ = 67.989, and t_{Byte} = 20.65 ms for N2, $\delta Error_{avg}$ =33.933 was obtained by simulation. Using this, nodes N1 and N2 sent 10 messages every 1 s for ten seconds. The message lengths were 5 and 15 bytes. For the two nodes, the results are summarized in table 1 for seven rounds. It was observed from the

Table 1. Synchronization Errors for Node 1 and Node 2

Round	Node	Sync Error (ms)	Node	Sync Error (ms)
1	1	41.39	2	99.43
2	1	29.82	2	95.55
3	1	41.34	2	91.72
4	1	37.50	2	91.72
5	1	37.52	2	95.56
6	1	41.34	2	95.56
7	1	37.52	2	95.56

simulation time that, as the number of bytes increased, the transmission time for a node increased, depending on number of bytes. The nondeterministic delay associated with size of a packet can be calculated as a correction factor (t_{Byte}* Number Of Bytes) and is applied before packet transmission.
- **Scenario 2 - Unequal distance between nodes:** Distance between N0 and N1 was 14.14 m, and between N0 and N2 was 26.92 m; the results obtained from this simulation were same as Scenario 1, with distance having less impact on nondeterministic latency. This can also be justified by the fact the travel times for radio waves remains nearly unvaried for short distances.
- **Scenario 3 - Synchronization error at different levels:** Suppose level i is the base station level, level $i+1$ is the child node levels, and level $i+2$ next child node levels up to level $i+n$, then the error at different levels can

be calculated relative to base station level as (8), where $tByte_{(i+n)-i}$ is the time required from a byte to be transferred between nodes of level n and i; N_i is the number of bytes to be transferred. The error at level $i+n$ is the cumulative error from level $i+n$ to i.

$$Error_{ni} = \sum_{n}^{i} tByte_{(i+n)-i} * N_i \tag{8}$$

4.2 Energy consumption

Energy consumed by a node in general is essentially due to two main contributors: (a) Processor - for processing data, storing data, issuing commands to Radio Interface (RI), etc and (b) Radio - to send, receive and perform operations on data received/sent. Energy consumed by a node varies depending upon the two main contributors (processor and RI) and also due to active and sleep times. During active times, the available energy is utilized (by processor and/or RI) up to or less than their stated absolute maximum rating values. During sleep (or idle) times, the nodes conserve their energy by entering into the low-power modes, in particular, sleep modes deactivate RI. Energy consumed by a node for the purpose of the communication, in general, at any particular instant, is given by (9); energy consumed by the main processor of the node is given by (10) and the total energy consumed by a node (at any instant of time) is (11)

$$E_{RI} = E_{Receiver} + E_{Sender} + E_{Radio\_Process} \tag{9}$$

$$E_P = E_{Process} * [T_{RadioON} + T_{RadioOFF}] + E_{Sleep} * T_{Sleep} \tag{10}$$

$$E_N = E_{Processor} + E_{RI} \tag{11}$$

From (9) - (11), for Imote2, E_{Pmax} = 97 mA; E_{Pmin} = 97 mA; E_{RImax} = 36.626 mA; E_{RImax}=27.5 mA. Energy expended in sending (E_{send}) and receiving ($E_{Receive}$) a packet are 31 mA and 18.1 mA respectively. For the cluster shown in Figure 2, energy consumed will be 32 w per cycle per cluster. For clusters with different spatial orientation, energy consumption entirely depends on number of clusters in the vicinity of each cluster head.

5 Conclusion

Time synchronization is fundamental to data fusion and henceforth to help detect and/or monitor events. The accuracy of synchronized time among nodes, coupling closely (in time), ameliorates the systems's ability to determine an event. Synchronization of time among nodes can be achieved by eliminating nondeterministic delays (send time, access time, propagation time, receive time). Adding to nondeterministic delays, frequency drifts due to clock instabilities and offset due to drifts, contribute to increase in synchronization error or mismatch in clock timings among nodes. Though several algorithms and approaches have been proposed by researchers for time synchronization, this proposed approach is also equally useful for energy-efficient, robust, externally synchronized multi-hop networks.

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-642-22543-7_32

References

1. Elson, J.: Time Synchronization in Wireless Sensor Networks. Ph.D. thesis, UCLA (2003)
2. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. In: Proceedings of the 5ht Symposium on Operating Systems Design and Implementation (OSDI '02). vol. 36, pp. 11–19. ACM (December 2002)
3. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03). pp. 138–149. ACM (November 2003)
4. Greunen, J.V., Rabaey, J.: Lightweight time synchronization for sensor networks. In: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA '03). pp. 11–19. ACM (September 2003)
5. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. vol. 2, pp. 1–10. IEEE (January 2000)
6. http://castalia.npc.nicta.com.au/: Castalia (Jan 2011 (verified on 6th Feb 2011))
7. Kim, B.K., Hong, S.H., Hur, K., Eom, D.S.: Energy-efficient and rapid time synchronization for wireless sensor networks. Consumer Electronics, IEEE Transactions on 56(4), 2258 –2266 (2010)
8. Kim, S.: Wireless Sensor Networks for Structural Health Monitoring. Master's thesis, UC-Berkely (2005)
9. Krishnamurthy, V., Fowler, F., Sazonov, E.: The effect of time synchronisation of wireless sensors on the modal analysis of structures. Smart Materials and Structures 17 (August 2008)
10. Maróti, M., Kusy, B., Simon, G., Ákos Lédeczi: The flooding time synchronization protocol. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04). pp. 138–149. ACM (November 2004)
11. Mills, D.L.: Internet time synchronization: The network time protocol. IEEE Transactions on Communications 39, 1482–1493 (October 1991)
12. Rmer, K., Blum, P., Meier, L.: Handbook of Sensor Networks: Algorithms and Architectures, chap. Time Synchronization and Calibration in Wireless Sensor Networks, pp. 199–237. Wiley and Sons (October 2005)
13. Schmid, T., Charbiwala, Z., Friedman, J., Cho, Y.H., Srivastava, M.B.: Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization. In: Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '08). pp. 97–108. ACM (June 2008)
14. Shahzad, K., Ali, A., Gohar, N.: Etsp: An energy-efficient time synchronization protocol for wireless sensor networks. In: 22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008). pp. 971–976 (2008)
15. Sichitiu, M.L., Veerarittiphan, C.: Simple, accurate time synchronization for wireless sensor networks. IEEE Wireless Communications and Networks (WCNC '03) 2, 1266–1273 (2003)
16. Sundararaman, B., Buy, U., Kshemkayani, A.D.: Clock synchronization for wireless sensor networks: A survey. Ad Hoc Networks 3, 281–323 (December 2005)
17. Wang, Y., Lynch, J.P., Kae, K.H.: A wireless structural health monitoring system with multithreaded sensing devices. Struct. Infrastruct. Eng. 3, 103–120 (2007)