TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 10, October 2014, pp. 7509 ~ 7522
DOI: 10.11591/telkomnika.v12i8.5759

Issues towards Efficient Time Synchronization in Wireless Sensor Networks

Abdul Waheed Khan, Abdul Hanan Abdullah*, Javed Iqbal Bangash
Faculty of Computing, Universiti Teknologi Malaysia UTM, Skudai, 81310, Johor Malaysia
Tel.: +607-553-8761; Fax: +607-553-8822
*Corresponding author, email: hanan@utm.my

Abstract

Wireless sensor network (WSN) is considered as the enabling technology to increase coordination between the physical and virtual worlds. A typical WSN is composed of a large number of computing devices known as nodes which are responsible for sensing and reporting some phenomenon to a sink or base-station where some useful conclusions are drawn from the reported data. Most of the times, the sensed data is of limited usage if not accompanied with timestamp and position information. In addition, several other basic operations in WSNs are based on efficient time synchronization such as duplicate detection, data aggregation/fusion, energy management, transmission scheduling, and cryptography etc; therefore time synchronization plays a pivotal role in operational activities of the network. In this paper, we discuss all the related issues and challenges that revolve around time synchronization in WSNs. Our purpose in writing this paper is to provide an in-depth understanding of the clock synchronization problem in WSNs so that the application designers can fine tune their applications in accordance with the underlined constraints and complexities involved.

Keywords: *wireless sensor networks, time synchronization, synchronization issues, clock drift*

1. Introduction

Wireless sensor network (WSN) is considered as the enabling technology to increase coordination between the physical and virtual worlds. A typical WSN is composed of a large number of computing devices known as nodes embodying limited set of resources such as a microcontroller, short range radio transceiver, and a long-lasting battery as energy supply (typically AA battery). These nodes are interfaced with sensors whose job is to sense and monitor the surrounding environment for various phenomenon such as temperature, sound, pressure etc and disseminate their sensed data to some special computing devices called sinks or base-stations in a coordinated manner where the sink nodes further process and analyze the reported data to draw conclusions about the reported activity [1]. In fact, WSN is considered as a special category of ad-hoc network which is characterized by the de-centralized infrastructure-free operating mode where nodes self-configure themselves upon deployment and carryout dual jobs of sensing and forwarding each other's data in a coordinated manner thus forming a multi-hop communication setup. However, nodes in WSN have their own unique characteristics that distinguish them from ad-hoc networks. Typical characteristics of WSN are limited energy resource, large scale deployment, cheaper but unreliable nodes and long operating/duty time. WSNs have got numerous applications and to name a few they have been very successful in enemy intrusion detection, precision agriculture, traffic control, infrastructure and machine health monitoring and patient's remote health monitoring. In short, they are particularly useful in situations where terrain, climate, and other environmental constraints hinder in the deployment of traditional wired networks [2].

The rest of the paper is organized as follows: In section 2, we discuss the importance of time synchronization in sensor networks and outline several basic operations in sensor networks which necessitate a common notion of time among the sensor nodes. To cope with the time synchronization problem, one needs to understand first the clocking mechanism in computing devices and how time inconsistencies over the network arise with the passage of time, an overview is presented in section 3. Taking into account the low cost characteristic of sensor nodes in WSN, the only cost-effective solution to attain clock synchronization in WSN is

Received February 8, 2014; Revised July 26, 2014; Accepted August 14, 2014

through the exchange(s) of time-synch messages thereby enabling the nodes to get an estimate of the remote clock's reading. However various components contribute to the non-deterministic communication latency, which potentially degrades the time synchronization process and are discussed in section 4. Section 5 discusses the three different message dissemination approaches being adapted for time synchronization in sensor networks. In section 6, we outline the typical features that need to be incorporated in a time synchronization scheme being developed for WSN. Various issues that arise in pursuit of adhering to the various features of WSN are being discussed in section 7. Section 8 discusses various schemes that have been developed to cope with time synchronization issues in WSN. Finally, we conclude our discussion with some recommendations in section 9.

2. Importance of Time Synchronization in Sensor Networks

In WSN, only the sensed data if not accompanied with timestamps and positions (sometimes), is of limited usage, therefore time synchronization plays a pivotal role to achieve the objective(s) the sensor nodes are deployed for. Several other basic operations in WSN are also based on efficient time synchronization. Time or clock synchronization is a mechanism that enable the sensor nodes in a network to correspond to a consistent notion of time either locally significant or globally (provided if a global time/clock reference is available). In the following lines, we briefly describe few such operations where efficient time synchronization is of immense importance:

Duplicate Detection and Data Aggregation: Nodes in sensor network are densely deployed and there is a high probability that the same event might be reported by multiple nodes operating in neighborhood. Since computation consumes very less energy compared to communication and therefore to avoid duplicate deliveries of an event, data aggregation mechanism is used where through the use of function like average, max, or min, transmission redundancy is eliminated from the data coming from multiple sources [5]. This requires accurate time-stamping.

Data Fusion: Sensor nodes are expected to coordinate with each other to achieve a complex sensing task. To do so, data fusion mechanism is employed to agglomerate raw data from nodes into some meaningful result forwarded enroute to the sink [6]. As an example, in a mobile target tracking application, different nodes report the time and location of the target to the sink node, from which the sink estimates the location and velocity of the target. Therefore, if the reporting nodes are not synchronized, it will result in an inaccurate estimate by the sink.

Energy Management: Nodes in sensor network are battery operated and usually there is no battery replacement service. Therefore to prolong network lifetime, all activities of sensor nodes must take into account the energy conservation goal. It has been well established that communication module in sensor node is the main consumer of the available energy resource and consumes more than 2/3 of energy resource [2]. Energy conservation schemes strongly depend on time synchronization. To avoid idle listening, duty-cycles are employed where nodes periodically go into sleep and wake-up modes and thus saves huge amount of energy by spending minimal energy in sleep mode. If these sleep and wake-up time intervals among the sensor nodes are not synchronized, it will result in slow delivery of sensed data because the neighbors might be asleep and thus unable to relay the data. Hence, network-wide synchronization is crucial for efficient duty-cycling that will ensure that although some nodes go into sleep mode, others in the neighborhood are still doing the sensing task and thus will make an efficient utilization of the energy resource.

Transmission Scheduling: MAC layer scheduling schemes such time division multiple access (TDMA) requires tight synchronization among the sensor nodes otherwise collisions might occur if two nodes try to transmit using the same time slot due to their unsynchronized clock timing.

Cryptography: To ensure freshness and prevent replay attacks, authentication schemes strongly dictate the use of synchronized time among the nodes.

Other than these, logging and debugging, localization and coordinated actuations also require a consistent notion of time among the sensor nodes.

3. Clock Model and Clock Inaccuracies

All computing devices are equipped with some clocking mechanism. A computer clock is comprised of a hardware oscillator (typically quartz-oscillators) and a counter that decrements its value with each oscillation of the quartz crystal. Whenever, the counter value gets to zero, an interrupt *(clock tick)* is generated which causes the software clock (another counter) to increment its value. It is the software clock that is read by applications through the use of some application programming interfaces (APIs). A software clock directly corresponds to the *local time* of a sensor node, and is represented by **C(t)** indicating the clock reading at some real time **t**.

Next we present some terminologies related to clocking that will help the readers to understand the inherent problems in clock hardware and will provide a better insight to cope with these issues.

3.1. Clock Offset

Two clocks might not correspond to the same time instant at the same time. Clock offset is the difference between the instantaneous absolute values of the clocks i.e., if two nodes **A** & **B** have the absolute clock values **C_A(t)** and **C_B(t)** respectively at a particular instance of time t, then clock offset can be represented by:

$$\theta_{AB} = C_A(t) - C_B(t) \tag{1}$$

For two perfectly synchronized clocks, clock offset is equal to zero. However, it is very rare in WSN either because different initial times might have been set by the manufacturers or the clock oscillation rate might change with passage of time subject to varying environmental conditions such as temperature, supplied voltage/pressure and aging effect.

3.2. Clock Frequency/Rate

It is the rate at which the clock progresses. The frequency of a node **A** at time t is given by:

$$C'_A(t) = \frac{d(C_A(t))}{d(t)} \tag{2}$$

Two nodes might be operating on different frequencies either because different initial frequencies might have been set by the manufacturers or the frequencies might change subject to the varying environmental conditions stated above. Frequency of clock also depends on type of crystal being used where some crystals are more stable compared to others in presence of varying environmental conditions.

3.3. Clock Ratio

It is the frequency ratio between clocks of two nodes. The ratio of clock of node **A** relative to node **B** at time t is given by:

$$\alpha = \frac{C'_A(t)}{C'_B(t)} \tag{3}$$

3.4. Clock Skew

It is the difference in frequencies of two clocks. Clock skew is the difference between the rates the clocks run i.e., the rate of change of offset. It is also defined as the rate of deviation of a node's clock from true time. Mathematically, it is the first order derivative of clock offset and is represented by:

$$\delta_{AB} = C'_A(t) - C'_B(t) = \frac{d(C_A(t))}{d(t)} - \frac{d(C_B(t))}{d(t)} \tag{4}$$

If two clocks are perfectly synchronized, then clock skew will be zero because the same rate of change of frequencies at both nodes' clocks will cancel out each other.

3.5. Clock Drift

Two synchronized clocks have a clock rate **dC/dt = 1** at all times but various factors affect the actual clock rate such as temperature and humidity of the environment, supplied voltage, and age of the quartz. As a result of this deviation, drift rate occurs which is the rate by which two clocks can drift apart i.e., **dC/dt – 1**. This drift rate is a variable quantity and the maximum drift rate of a clock **(ρ)** for quartz-based clocks might take a value from the range 1 part-per-million (ppm) to 100 ppm where 1 ppm = 10-6. The value of **ρ** is usually given by manufacturer of the oscillator and it must guarantee the following relationship:

$$1-\rho \leq \frac{dC}{dt} \leq 1+\rho \tag{5}$$

Based on the variations in a clock's drift rate with respect to real clock time, we might have fast, perfect, or slow clocks as shown in Figure 1.

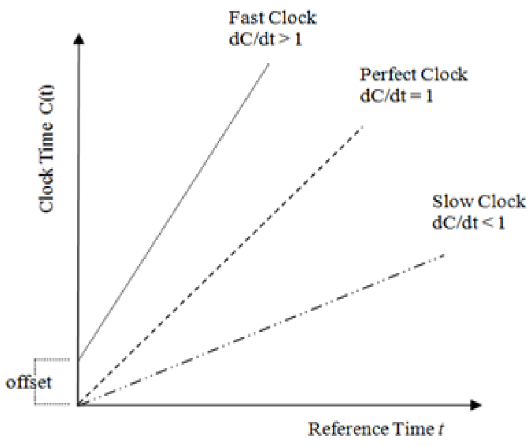

Figure 1. Classification of Clock Based on Drift Rate

Sensors' clocks might get synchronized at one instant of time but due to clock skew and consequently due to the drift rate the sensors' clock readings might become inconsistent at a later stage, making it necessary to repeat the synchronization process to achieve a common notion of time. Even in the absence of clock skew, two nodes might not have consistent time notion possibly due to a slightly different initial frequency set by the manufacturer. Now the question arises that how long this resynchronization interval should be? To find out this interval, consider two identical synchronized clocks which can drift apart from each other at a rate of at most 2ρmax (if one clock drifts positively and the other drifts negatively), now to bound the relative offset to δ seconds, the resynchronization interval **(τ_{sync})** must satisfy the following requirement:

$$\tau_{sync} \leq \frac{\delta}{2\rho_{\max}} \tag{6}$$

It is a norm to make gradual clock adjustments, for example, using a linear compensation function that changes the slope of the local time, otherwise simply jumping forward or backward would result in missing or repeating the time-triggered events [3].

3.6. Clock's Precision and Stability

These are the two important parameters of a clock's oscillator in a computing device. Precision (a.k.a frequency error) is the difference between theoretical and real frequencies of an oscillator and is usually given by manufacturer in the units of ppm. Essentially, it is the drift rate of a node's clock with respect to an ideal and perfect clock. Clock stability is the tendency of the clock's oscillator not to deviate from the same frequency over the time and is affected by the factors like temperature and humidity, supplied voltages, pressures and material aging.

3.7. Adjusting Drifted Clocks

Complications arise when the local clock of a node runs faster than the clock of reference node. If $\mathbf{T_{curr}}$ represents the current local clock time and $\mathbf{T_{new}}$ represents the real time that the local clock has to be updated to, then:

a) If $\mathbf{T_{new} > T_{curr}}$ then the software time of the node is simply advanced to the new value Tnew.
b) If $\mathbf{T_{new} < T_{curr}}$ then we cannot directly set back the software time of the node to the new value $\mathbf{T_{new}}$ because doing that will result in faulty timestamps (some time-triggered events will be repeated). To cope with this issue, instead of tuning the clock back, the clock is slowed down by the software that handles the clock tick interrupt until it gets to a desired value progressively.

3.8. Complications in Clock Offset and Clock Skew Estimation

The problem of clock synchronization requires accurate estimates of both the clock offset and clock skew (if any). Synchronization schemes that correct only the clock offset [4], requires frequent resynchronization than those that correct both the clock offset and skew. The reason behind this frequent resynchronization is that clocks start drifting apart in case of uncompensated clock skews. Therefore, joint clock offset and skew compensation results in long term reliability of synchronization and thus more energy conservation. If clocks of two nodes are perfectly synchronized, then clock offset is zero and the difference between the *Send Timestamp* and *Receive Timestamp* will give us the end-to-end delay. In case of non-zero offset but no clock skew, the end-to-end delay will be the difference of the two timestamp plus the clock offset. The situation gets complicated in case of non-zero clock offset and non-zero clock skew because the end-to-end delay might gradually increase or decrease over time subject to whether the sender clock runs slower or faster compared to receiver clock.

4. Non-Deterministic Communication Latency in Wireless Environments

In order to achieve a common and consistent notion of time, the nodes exchange their local clock readings with each other to estimate and adjust their clock parameters. However, various factors affect the message delivery in a wireless network and complicate the synchronization process. In fact, there is no guarantee that each receiver will receive the transmitted signal at the same instant. This variable communication latency contributes to unpredictable delays. Some of the factors that contribute to communication latency while transmitting over a wireless channel are *Send Time, Access Time, Transmission Time, Propagation Time, Reception Time, and Receive Time* [7] and [4] and the combined effect of all these different kinds of delays must be less than the required tolerance of the time synchronization. In the following lines, we provide a brief description of the various delays in message delivery:

Send Time: The time to construct the message and then pass on to MAC layer on the transmitter side. It might include delays caused by operating system such as context switching, system-call overhead, and the processor's current load. All these components of delays are of variable length and cannot be precisely predicted. Putting simply, it is the time required to transfer the message from application layer to network interface.

However, the delay caused by Send-Time can be eliminated from the overall end-to-end delay by using MAC layer time-stamping as being done in TPSN [4], FTSP [14], DMTS [20] etc.

Access Time: The time that a message must wait before getting access to the wireless medium/channel. Since wireless medium is a shared medium and if multiple nodes try to transmit at the same time, collisions are inevitable. To avoid such scenarios, two categories of

MAC protocols exist: Contention-based MAC protocols and Schedule-based MAC protocols. Contention-based MAC protocols need an exchange of control messages Request-to-Send (RTS) and Clear-to-Send (CTS) to make sure that the medium is free before transmission. In case of unavailability, the node waits for a random amount of time. In schedule-based MAC protocols such as TDMA, every node is assigned a time-slot when it can get access to the medium so the node waits for its time-slot before transmission. The Access-Time is the least deterministic as it strongly depends on the density of nodes and the network traffic. Simply putting, it is the waiting time to access the transmission channel until the transmission begins.

Again, by using the MAC layer time-stamping, the Access-Time delay can be eliminated from the end-to-end delay as being done in TPSN [4], FTSP [14], DMTS [20], etc.

Transmission Time: The time that it takes for the sender to transmit the message once it gets access to the wireless medium. It depends on the length of the message and the speed of the radio (the time that it takes to transmit one bit of the message).

This component of delay is partly deterministic if the length of the message is known in advance. In that case, the Transmission-Time is simply **nT**, where **n** is the number of bits in the message and **T** is the time to transmit one bit of the message.

Propagation Time: Is the actual propagation time for a message through the wireless medium once it lefts the sender. Message in a wireless channel travels with speed of light and it depends on the distance between two nodes.

In wireless sensor networks, delay caused by Propagation-Time is usually negligible taking into consideration that nodes are only a few meters apart.

Reception Time: The time required for the receiver to receive the message at the physical layer. It is just like Transmission-Time and depends on length of message and speed of the radio.

Receive Time: Is the time for the network interface card to receive, construct the message at the receiver and notify the host of its arrival.

Receive-Time does not contribute much to end-to-end delay of message delivery and can easily be eliminated by MAC layer time-stamping at the receiver end.

5. Time-Synchronization Message Dissemination Approaches

There are typically three approaches for exchanging time-synchronization messages to achieve clock synchronization in wireless sensor networks: first one is one-way time-synchronization message dissemination, second is two-way message exchange (a.k.a sender-receiver synchronization) and finally the receiver-receiver synchronization [8]. To achieve network-wide synchronization, this process of exchanging timing messages is repeated among multiple node pairs until all nodes in the network adjust their local clocks. Here, we provide a critical analysis of these message dissemination mechanisms towards time synchronization.

5.1. One-Way Message Dissemination

This is a simple pairwise message exchange in which two nodes synchronize their clocks using only a single message. Lets clocks of two nodes **i** and **j** needs to be synchronized using this approach. As illustrated in Figure 2, the synchronization process begins when node **i** sends a synchronization message to node **j** at time $\mathbf{t_1}$ and time-stamps that message with $\mathbf{t_1}$. Upon receiving this message, node j timestamps this message with time $\mathbf{t_2}$ according to its local clock reading and from the difference of these two timestamps, the clock offset θ is estimated, that is:

$$\theta = t_2 - t_1 \tag{7}$$

Accordingly, node j adjusts its clock time by the following equation:

$$t_2 = t_1 + D + \theta \tag{8}$$

Where **D** is the propagation delay of the message and since nodes in sensor network are only a few meters apart, so it is either ignored or is assumed to be a certain constant value.

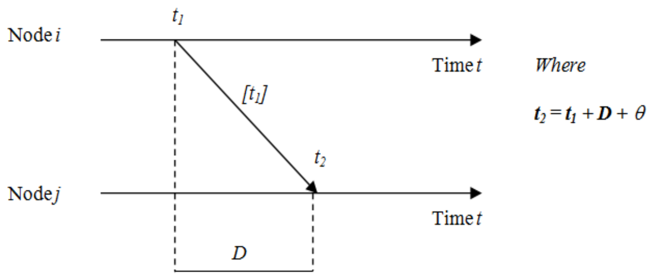

Figure 2. Clock Synchronization using One-way Message Dissemination

This approach is although very simple but due to various delay components mentioned above especially the *media access delay* which is highly non-deterministic (depending on network traffic) and is normally several order higher in magnitude than the clock period, one single time transfer would not be enough to synchronize the nodes. Another issue with one-way message dissemination is that the sender clock skew might change after it timestamps and sends the synchronization message to the receiver. The receiver has no means to learn about this change in sender's clock skew other than if the sender sends another message to receiver notifying about its change of the clock skew which adds to the communication overhead and thus can potentially consume more energy resource.

5.2. Two-Way Message Exchange

This approach is also known as round-trip synchronization and is somehow more accurate approach as shown in Figure 3. Using this approach, node **i** first timestamps the time-synch message with time t1, which is received and recorded by node **j** at time **t_2**.

In return, node **j** sends a response to node i timestamped with time **t_3** and also including **t_1**. Upon receiving this response, node **i** time-stamps the message with time **t_4** which is computed using the following equation:

$$T_4 = t_3 + D + \theta \tag{9}$$

If the communication latency is symmetric, then:

$$D = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{10}$$

And,

$$\theta = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{11}$$

Substituting values of **D** and θ in Equation (9), node i will adjust its clock timing accordingly. Now if node **i** has to notify to node **j** about the estimated offset, it can do so by transmitting a third message to node **j**.

This approach works well under the assumption that clock offset is constant during the exchange process and clocks do not drift much in this entire exchange process which may not be true because clock offset continuously grows subject to different frequencies of the oscillators. Another big assumption is that the communication latency is symmetric and remains constant.

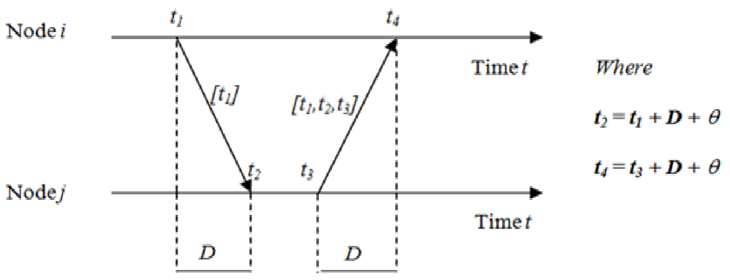

Figure 3. Clock Synchronization using Two-way Message Exchange

5.3. Receiver-Receiver Message Exchange

Using this approach, a group of nodes receive the same message broadcasted by a reference node and therefore, it is also known as reference broadcasting. As illustrated in Figure 4, the broadcasted message is not time stamped by the sender, rather upon receiving, all the receiving nodes timestamp it according to their local clock times and then exchange their arrival times to compute an offset which would be the difference in reception times. In case if there are only two receivers, then a total of three messages would be required to synchronize these two receivers. Reference broadcast scheme (RBS) [9], follows this approach to synchronize all the nodes that are in the radio coverage of the sender.

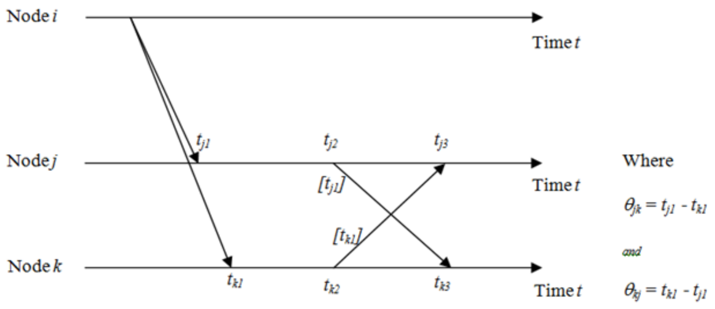

Figure 4. Clock Synchronization using Receiver-receiver Message Exchange

Offset computation using this approach, is more accurate compared to two-way message exchange since the broadcasted message does not encounter random delays at sending node. However, it incurs significant time uncertainty because the message reception takes some time (e.g., subject to receiver's radio speed, processor availability etc) from one node to another, hence there might be significant difference in the absolute values of the relative offset computed at any pair of receivers. Another issue in this approach is that the sending node remains unsynchronized in this process.

6. Features of Time Synchronization Schemes in Wireless Sensor Network

The scarce resources of nodes in WSN make it very difficult for any time synchronization scheme to attain an optimal synchronization without compromising some of the ideal characteristics. Elson and Estrin [10], have identified a set of metrics for time synchronization in WSN that serves as a benchmark for any time synchronization scheme being developed for WSN. However, no single scheme is considered optimal along all the axes and trade-offs are always being observed between the various metrics (for example, an accurate time synchronization would be attained at the expense of more energy consumption). In the following lines, we briefly describe these features that need to be incorporated in a time synchronization scheme being developed for WSN:

Energy Constraints: Energy is the scarcest resource in WSN, therefore like all other operations in WSN, the synchronization scheme should take into account the limited available energy and therefore should opt for least energy consumption. Some traditional time synchronization schemes require the use of sophisticated and energy-hungry equipments (e.g., GPS receivers and atomic clocks), however the use of such equipments in WSNs is not a viable option due to the energy consumption and cost issues. Time synchronization in WSN is achieved through exchange of time-synch messages; therefore an efficient time synchronization scheme should aim for achieving time synchronization at the expense of minimum number of communication messages. However, there is a trade-off between energy efficiency and synchronization accuracy and limiting the messages result in more synchronization error.

Accuracy: Accuracy is a measure of how close a node's time is to the true time. The accuracy of time synchronization is highly application specific. In certain applications, the synchronization accuracy might be in the order of a few ***µsec*** while in others only a simple ordering of events or messages would be sufficient. High accuracy can be achieved through the use of high frequency oscillator but at the expense of more energy consumption.

Computational Complexity: Nodes in WSNs have limited hardware capabilities and have severe energy constraints; therefore the complexity (run-time and memory requirements etc.) of a time synchronization protocol can make a protocol impractical for many applications if its computational requirements exceed the node's physical resources.

Scalability and Mobility Support: In many applications of WSN, nodes are deployed on a large scale. Further due to mobility, frequent topological changes might happen and since communication interference depends on density of network, therefore the synchronization scheme should be able to accommodate the increasing number of nodes and scale well accordingly without degrading synchronization accuracy.

Robustness: Nodes being battery operated might deplete their batteries but the synchronization scheme should remain functional in case of these topological changes.

Cost and Size: Small size and low cost are the main characteristics of nodes in WSNs towards their widespread adaptability. Nodes cannot afford to have large and expensive hardware (such as GPS or temperature compensated clocks) attached to them. Therefore, lightweight synchronization schemes should be developed in accordance with the limited cost and small size of the sensor nodes.

7. Issues and Challenges towards Efficient Time Synchronization

In this section, we outline the various issues being faced while adhering to some of the typical characteristics of WSN in pursuit of time synchronization.

7.1. Issues in Energy Efficient Time Synchronization

Energy efficient synchronization can be attained by adhering to two basic principles in clock synchronization process [11]:

1. Using Low Frequency Clock
2. Infrequent Communication

Hennessy and Patterson, have identified the following relationship between the power and frequency in [12] and is considered as rule of thumb for embedded systems.

$$Power = Capacitative\ Load(C) * Voltage^2 * Frequency\ Switched \quad (12)$$

Therefore, low-frequency clocks are mandatory to minimize the power dissipation of the oscillator, digital counter and clocking network. However, a low-frequency clock compromises the time resolution and frequency error thereby resulting in less finer time resolution and more frequency error, for example, an 8MHz clock results in frequency error resolution of 0.0125 ppm, while a 32KHz clock results in 3.05ppm [11].

Communication module on the other hand is considered as the largest consumer of the node energy resource, for example: the energy required to transmit 1 bit over 100m can be used to execute 3 million instructions [13]. Therefore, an important characteristic being sought towards energy conservation is less communication overhead during synchronization process. However, less or infrequent communication results into larger synchronization intervals which compromises the accuracy of time notion among the nodes in the network because changing the environmental temperatures or exposure to high voltages would cause greater frequency error (clock skew) and thus high drift rate between the resynchronization intervals. Frequent exchange of synch messages results in fast convergence and increases the accuracy at the expense of more power consumption so there is a trade-off between energy consumption, accuracy and convergence time [14] whereby decreasing the resynchronization interval increases the number of time-synch messages sent in a certain time period.

7.2. Issues in Improving Accuracy of Time Synchronization

As stated earlier, the accuracy of time synchronization is highly application specific where some applications might require accuracy of time synchronization in ***µsec***, others might require a few ***msec*** while in some applications simple ordering of events would do the job. There is a trade-off between the accuracy and energy consumption whereby higher accuracy is achieved at the expense of more energy consumption thereby achieving faster convergence through frequent exchange of synch messages [14]. To conserve energy during the synchronization process, an alternate approach is to exchange the time-synch messages initially over some time interval for estimation of clock offset and skew and once sufficient data points are collected then use statistical techniques such as linear regression to predict clock offset using the best-fit line as shown in figure 5. If the oscillator frequencies of WSN nodes do not fluctuate, then a one-time estimation of clock drift would be sufficient to transform the clock reading of another node. This approach helps in reducing the exchange of time-synch messages and thus conserving energy. However, these statistical techniques are not guaranteed to come up with the right estimates of clock parameters and thus the synchronization accuracy degrades significantly especially in case of unstable conditions such as ambient temperature variations or changing the supplied voltages where the clock skew will drift far away than expected. Therefore, in case of changing environmental factors, the synchronization procedure needs to be repeated after some time.

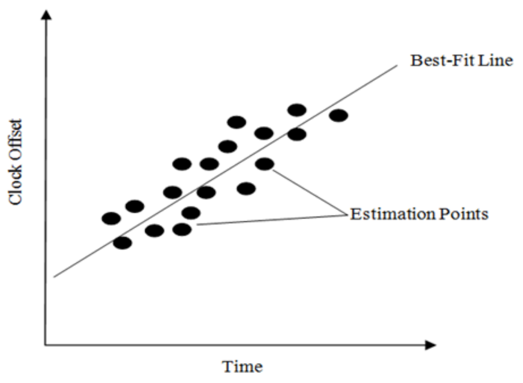

Figure 5. Clock Offset Predication using Best-Fit Line

7.3. Issues towards Scalable Time Synchronization

An important characteristic of WSN is their self-organization and dynamic behavior which implies that there might be topological changes from time to time that need to be accommodated by the time synchronization schemes. In essence, if new nodes join the network and the network size grows, the time synchronization scheme should adapt itself accordingly and provide the synchronization services to the newly joining nodes, however the synchronization scheme should opt for less degradation of accuracy. In WSN, nodes can report data to sinks either in a flat-based (peer-to-peer) or hierarchical fashion and accordingly the time synchronization schemes can be flat-based or hierarchical. Compared to peer-to-peer synchronization, scalability is easily achieved with less overhead by hierarchical and cluster-based time synchronization schemes such as TDP [15], FTSP [14], and TPSN [4, 16] etc., as they keep a provision for topological changes and offer the flexibility to adapt accordingly. In such synchronization schemes, new nodes might join a network from two different points: new nodes enter into the direct coverage area of ring/cluster leader; nodes join the network at a point which is outside the coverage area of ring/cluster leader but they are in the vicinity of border synchronized nodes - being the children nodes of the ring/cluster leader. In the former case, the new nodes easily get synchronized by receiving direct time-synch messages being periodically broadcasted by the ring/cluster leader, where as the latter case requires that the newly joined nodes need to form another level of hierarchy first thereby making themselves children of an already synchronized node and then repeat the same process for synchronization just like their parent node had done so. The whole network gets synchronized in this way, however, the synchronization error increases per hop as the network size grows because nodes in lower levels will only get synchronized once their parent nodes get themselves synchronized with upper levels in the hierarchy.

7.4. Issues towards Robust Time Synchronization

Robustness is another important characteristic being sought in time synchronization schemes where the synchronization schemes should remain functional in case of topological changes being triggered by nodes' mobility or depletion of energy resource. In peer-to-peer synchronization, robustness can easily be achieved as any node can communicate directly to any other node in the neighborhood but it becomes a serious problem in master-slave synchronization where the master node's failure could disrupt the whole synchronization process as all the slave nodes get their clocks synchronized by listening to master node's time-synch messages [17]. In master-slave synchronization, master nodes periodically broadcast time-synch messages and no such reception of these messages by the slave nodes over a certain time interval lead them to the conclusion that a topological change has been occurred and a re-election process needs to be carried out for the new master node such as TPSN [4], FTSP [14], and TDP [15]. Therefore, in master-slave synchronization, robustness comes at the expense of more delay in convergence time, more network traffic, and more energy consumption.

8. Contribution of Existing Time Synchronization Schemes

Several time synchronization schemes have been developed by researchers where some schemes aim at time synchronization in computer networks while others are specifically developed for time synchronization in sensor networks. In the following sections, we discuss why the traditional time synchronization schemes cannot be applied in WSNs, followed by an overview of state-of-the-art time synchronization schemes being explicitly designed for WSN.

8.1. Why Traditional Time Synchronization Methods Not Applicable in WSNs?

Many protocols have been designed to maintain synchronization of the physical clocks over the computer networks. These traditional methods are based on Network Time Protocol (NTP) proposed by Mills [18] and GPS. NTP although being very successful in traditional infrastructure-supported networks but the availability of CPU-cycle all the times, listening to network all the times, access to global time-scale via GPS, and centralized operational modes are some of the assumptions that hold true in traditional networks, but are not applicable in WSNs due to the scarce resources [19]. Furthermore, in traditional networks, a GPS receiver provides an external persistent, global clock but due to energy and cost limitations, the use of such receivers is not a viable option in WSN environments.

8.2. State-of-the-art Time Synchronization Schemes for WSN

In this section we discuss several candidate time synchronization schemes being specifically proposed and designed for WSN environments. In essence, we discuss: how these candidate time synchronization schemes cope with the stringent characteristics being posed by WSNs; what do they get and lose in this race.

Reference Broadcast Scheme (RBS) proposed by Elson et al. [9] is based on receiver-receiver message exchange mechanism for time synchronization in WSN. RBS focuses only on accuracy of time synchronization and achieves accuracies in the order of few ***µsec***. Since it was designed for networks with small number of nodes, therefore gives poor performance in terms of energy consumption and accuracy when it comes to large scale deployment of nodes i.e., energy consumption is dependent on density of nodes in a given area. To synchronize a large scale network, it would require an unnecessarily large number of transmissions which would quickly deplete the sensor’s energy resource. Furthermore, due to energy depletion, nodes would go down and network connectivity would not be maintained, resulting in reduced network coverage.

Timing-Synch Protocol for Sensor Networks (TPSN) proposed by Ganeriwal et al. [4] achieves a network-wide time synchronization using two-way message exchange mechanism. The authors claim a two times better performance in terms of accuracy when compared to RBS on Berkeley motes platform. TPSN creates a hierarchical topology structure in the network thereby forming different levels to reduce complexity for multi-hop communication, and then pair-wise synchronization is performed between a pair of nodes along the edges of the hierarchical tree. Synchronization is initiated by a root node (level 0) by broadcasting a time-synch packet where level 1 nodes upon receiving the packet enter into a two-way handshake with the root node and in this way, get their clocks synchronized with the root node. This process is repeated down to the lowest levels in the hierarchy where nodes in level **i** get their clocks synchronized with some node in level **i-1**. Compared to RBS, TPSN is more energy efficient (fewer transmissions are required for synchronization), accurate and scalable as the accuracy is less deteriorated with increase in network size. However, the overall energy conservation is not very effective as physical clock corrections are performed rather than remote clock’s transformation. TPSN being a hierarchical tree based scheme, is also not very suitable for applications where there is high node mobility as that would result in frequent topological changes which results in more energy consumption.

Delay Measurement Time Synchronization for Wireless Sensor Networks (DMTS) [20] is another sender-receiver synchronization scheme where one sender synchronizes multiple receivers at the same time using one-way message dissemination mechanism. It avoids the round-trip time estimation and therefore requires fewer transmissions to get the network synchronized. It also allows multi-hop synchronization thereby upon detecting the child nodes, the already synchronized nodes broadcast their time signals. DMTS exhibits quite low computational complexity (no complex mathematical operations are involved) and achieves high energy conservation (since only a single message is required to synchronize all nodes within the coverage limits of the transmitter), however synchronization accuracy is degraded in return and is only suitable for applications where synchronization accuracy is not a major concern.

Flooding Time Synchronization Protocol (FTSP) [14] attains a network-wide synchronization by using multi-hop synchronization. It works much the same way as TPSN, however unlike TPSN, it is based on one-way message dissemination. To reduce the number of transmissions, FTSP employs linear regression technique to compensate clock drifts that might occur with passage of time. In FTSP, the root node (a dynamically elected node) keeps a global time. All other nodes in vicinity of root node get their clocks synchronized by making use of a single radio message broadcasted by root. To deliver the root’s global time to other nodes which are outside the radio range of root node, the already synchronized nodes form an adhoc structure from root node to all other nodes as opposed to fixed spanning tree in TPSN. In this way, it avoids the need for having an initial phase for tree establishment and thus provides mobility-support and robustness against node/link failure. On small scale networks, FTSP achieves good global synchronization at low communication cost; however it incurs large skews between neighboring nodes due to large stretch of the employed tree structure [21].

Lightweight Tree-based Synchronization (LTS) proposed by Greunen and Rabaey [22] unlike most other time synchronization schemes, does not focus on maximizing the accuracy of synchronization but rather aims at minimizing the complexity of the synchronization process. The authors claim that most of the times, the maximum desired accuracy in sensor networks is relatively low (within fractions of seconds) and synchronization scheme for WSN should be lightweight in order to reduce the complexity and prolong the network lifetime.

Consensus Clock Synchronization (CCS) recently proposed by Maggs et al. [23], aims to achieve a network wide clock synchronization not with respect to an external time source like UTC but with internal consensus in the network on what time is, and how fast it travels. The network clocks converges progressively to a consensus time by going through various rounds where at the end of each synchronization round, the scheme updates the compensation parameters (clock offset and skew) for each node. CSS is based on a consensus clock which is not a physical clock but a virtual clock that is generated from those nodes in the network running the CSS algorithm and accordingly it has its own skew rate and offset relative to the absolute time. CSS compensates both offset and skew in each round where in the offset compensation phase, nodes exchange local clock readings to achieve a common time; for skew compensation, the skew compensation parameters are improved by iteratively comparing the results from current and previous synchronization rounds. The synchronization rounds are repeated every $\mathbf{t_{sync}}$ where the value of $\mathbf{t_{sync}}$ can be adjusted according to application requirements. CSS although improves the accuracy of time synchronization and confidence level since all nodes transmit and participate in each synchronization round but in case of densely deployed nodes, the traffic generated per synchronization round would be enormously increased thereby resulting in more energy consumption and long convergence time (as nodes can only broadcast in their turns which is determined by MAC protocol to avoid any conflicts over the wireless medium).

9. Conclusion and Recommendations

Among the many challenges posed by resource constrained WSNs, time synchronization is of critical importance as it not only helps in energy conservation but also provides the basis for several other basic operations in sensor networks. Time synchronization problem in WSN has been extensively studied in literature over the last decade, however there is no single time synchronization scheme which is at the same time energy efficient and achieves high degree of accuracy with support of scalability—independent of the underlying topology. It is very difficult to design such a time synchronization scheme which has maximum efficiency along all the axes—energy efficiency, accuracy, scalability, and computational complexity etc. because there are always trades-off among these parameters. After analyzing the performance of several existing time synchronization schemes towards solving the synchronization issues, we recommend that instead of designing different time synchronization schemes for different application environments, the time synchronization in WSNs should be provided as a flexible middleware solution which integrates the various synchronization features—energy efficiency, accuracy, scalability and computational complexity etc. The specific implementation should be left as an application designer's choice where they can tune their applications to specific synchronization parameters according to their application requirements.

References

[1] Lewis FL. Wireless Sensor Networks in Smart Environments: Technologies, Protocols, Applications. New York: John Wiley. 2004: 11-46.
[2] Halawani S, Khan AW. Sensors Lifetime Enhancement Techniques in Wireless Sensor Networks - A Survey. *Journal of Computing*. 2010; 2(5).
[3] Dargie W, Poellabauer C. Fundamentals of wireless sensor networks-theory and practice. John Wiley & Sons. 2010.
[4] Ganeriwal S, Kumar R, Srivastava M. *Timing-sync protocol for sensor networks*. Proceedings of the 1st ACM Conference on Embedded Network Sensor Systems. Los Angeles. 2003; 138–149.
[5] Heidemann J, Silva F, Intanagonwiwat C, Govindan R, Estrin D, Ganesan D. *Building Efficient Wireless Sensor Networks with Low-Level Naming.* 18th ACM Symposium on Operating Systems Principles, Banff. 2001; 146-159.
[6] Akyildiz F, Su W, Sankarasubramaniam Y, Cayirci E. Wireless Sensor Networks: A Survey. *Computer Networks*. 2002; 38(4): 393-422.
[7] Kopetz H, Ochsenreiter W. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers.* 1987; 36(8): 933–940.
[8] Wu YC, Chaudhari Q, Serpedin E. Clock Synchronization of Wireless Sensor Networks. *IEEE Signal Processing Magazine*. 2011; 28(1): 124-138.
[9] Elson J, Girod L, Estrin D. *Fine-grained network time synchronization using reference broadcasts*. Proceedings of the Fifth Symposium on Operating System Design and Implementation. Boston, MA. 2002; 36: 147-163.
[10] Elson J, Estrin D. *Time Synchronization for Wireless Sensor Networks*. Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS '01). San Francisco. 2001; 1965-1970.
[11] Schmid T, Dutta P, Srivastava M. *High-resolution, low-power time synchronization an oxymoron no more*. In ACM/IEEE IPSN. 2010.
[12] Hennessy JL, Patterson DA. Computer Architecture: A Quantitative Approach. 3rd Edition. San Francisco, CA: Morgan Kaufmann. 2002.
[13] Pottie G, Kaiser W. Wireless integrated network sensors. *Communications of the ACM*. 2000; 43(5): 51–58.

[14] Maroti M, Kusy B, Simon G, Ledeczi A. *The Flooding Time Synchronization Protocol.* Proceedings of the 2nd ACN Conference on Embedded Networked Sensor Systems (SenSys). Baltimore, Maryland. 2004; 39-49.
[15] Su W, Akyildiz IF. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*. 2005; 2: 384-397.
[16] Kim H, Kim D, Yoo S. *Cluster-based hierarchical time synchronization for multi-hop wireless sensor networks*. 20th International Conference on Advanced Information Networking and Applications, AINA'06. Vienna. 2006; 2(5): 18-20.
[17] *Rhee* IK, Lee J, Kim J, Serpedin E, Wu YC. Clock Synchronization in Wireless Sensor Networks: An Overview. *Sensors*. 2009; 9(1): 56-85.
[18] Mills DL. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications.* 1991; 39(10): 1482-1493.
[19] Elson J, Romer K. *Wireless Sensor Networks: A New Regime For Time Synchronization*. In Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I). Princeton, New Jersey. 2003; 33(1): 149-154.
[20] Ping S. Delay Measurement Time Synchronization for Wireless Sensor Networks. *Intel Research, IRB-TR-03-013.* 2003.
[21] Lenzen C, Sommer P, Wattenhofer R. *Optimal clock synchronization in networks*. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. Berkeley. 2009; 225-238.
[22] Greunen JV, Rabaey J. *Lightweight Time Synchronization for Sensor Networks.* Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA). San Diego, CA. 2003; 11-19.
[23] Maggs MK, O'Keefe SG, Thiel DV. Consensus Clock Synchronization for Wireless Sensor Networks. *IEEE Sensors Journal.* 2012; 12(6): 2269-2277.