TIME SYNCHRONIZATION IN WIRELESS SENSOR NETWORKS: A SURVEY

Prakash Ranganathan, Kendall Nygard
Department of Computer Science, North Dakota State University, Fargo, ND, USA
prakashranganathan@mail.und.edu

Received: 12th January 2019 Revised: 25th April 2019 Accepted: 16th May 2019

ABSTRACT — *Time synchronization is a critical piece of infrastructure for any distributed system. Wireless sensor networks have emerged as an important and promising research area in the recent years. Time synchronization is important for many sensor network applications that require very precise mapping of gathered sensor data with the time of the events, for example, in tracking and vehicular surveillance. It also plays an important role in energy conservation in MAC layer protocols. The paper studies different existing methods, protocols, significant time parameters (clock drift, clock speed, synchronization errors, and topologies) to achieve accurate synchronization in a sensor network. The studied Synchronization protocols include conventional time sync protocols (RBS, Timing-sync Protocol for Sensor Networks -TPSN, FTSP), and other application specific approaches such as all node-based approach, a diffusion-based method and group sync approaches aiming at providing network-wide time. The goal for writing this paper is to study most common existing time synchronization approaches and stress the need of a new class of secure-time synchronization protocol that is scalable, topology independent, fast convergent, energy efficient, less latent and less application dependent in a heterogeneous hostile environment. Our survey provides a valuable framework by which protocol designers can compare new and existing synchronization protocols from various metric discussed in the paper. So, we are hopeful that this paper will serve a complete one-stop investigation to study the characteristics of existing time synchronization protocols and its implementation mechanism in a Sensor network environment.*

Keywords: Secure-time, Synchronization, MAC Layer

1. INTRODUCTION

The time synchronization problem to synchronize the local clocks of sensor nodes in the wireless network have been extensively studied in literatures over the last two decades and yet there is no specific time synchronization scheme available to achieve higher order of accuracy with greater scalability independent of topology and application [1-4]. This reflects the complexity associated with the collaborative nature of sensor nodes, in many sensor networks causing it to be a trivial problem in itself. One of the basic middleware services of sensor networks is network-wide time synchronization within the network. Many sensor network applications require time to be synchronized within the network. Examples of such applications include environmental monitoring, mobile object (target) tracking, data fusioning, TDMA radio scheduling, message ordering, to name a few. Consider the application of mobile object tracking, in which a sensor network is deployed in an area of interest to monitor passing objects. When an object appears, the detecting nodes record the detecting location and the detecting time. Later, these location and time information are sent to the aggregation (sometime referred as fusion) node which estimates the moving trajectory of the object. Without an accurate time synchronization scheme, the estimated trajectory of the tracked object could differ significantly from the actual one. Hence, a precise either global (or localized) timesync service is critical and must be made available at each of the sensor nodes for any application. In this paper, we study all class of existing time synchronizing schemes and provide a comprehensive analysis to future researches.

2. TIME SYNCHRONIZATION PROBLEM

The time of a computer clock is measured as a function of the hardware oscillator

$$C(t) = k \int_{t0}^{t} w(T)\, d\, T + c(to)$$

where
$w(T)$ is the angular frequency of the oscillator, k is a constant for that oscillator, and t is the time. The change of the value $c(to)$ leads to the events (or interrupts) that can be captured by the sensor. The clocks in a sensor network can be inconsistent due to several reasons. The clock may drift due to environment changes, such as temperature, pressure, battery voltage, etc. This has been a research topic in the operating system and Internet communities for many years. There are three reasons for the nodes to be representing different times in their respective clocks – (1) The nodes might have been started at different times, (2) The quartz crystals at each of these nodes might be running at slightly different frequencies, causing the clock values to gradually diverge from each other (termed as the *skew* error), (3) The frequency of the clocks can change variably over time because of aging or ambient conditions such as temperature (termed as the *drift* error). All the above said errors are three sources that contribute to different time within a sensor network. The nodes in a sensor network may not be synchronized well initially, when the network is deployed. The sensors may be turned on at the different times and their clocks may be running according to different initial values. The results of events on specific sensors may also affect the clock. For example, the Berkeley Mote sensors may miss clock interrupts and the chance to increase the clock time value when they are busy handling message transmission or sensing tasks [1]. We will explore the time synchronization problem due to clock offset by analyzing some related work on existing time synchronization schemes in the next section.

3. RELATED WORK ON TIME- SYNC SCHEMES

The early time sync protocol used in the internet domain is the Network Time Protocol (NTP) devised by Mills [2]. The NTP clients synchronize their clocks to the NTP time servers with accuracy in the order of milliseconds by statistical analysis of the round-trip time. The time servers are synchronized by external time sources, typically using GPS. The NTP has been widely deployed and proved to be effective, secure and robust in the internet. In WSN, however, non-determinism in transmission time caused by the Media Access Channel (MAC) layer of the radio stack can introduce several hundreds of milliseconds delay at each hop. Therefore, without further adaptation, NTP is suitable only for WSN applications with low precision demands [2]. There were other schemes such as Reference Broadcast schemes proposed by Elson et al. [3] which eliminate the uncertainty of the sender by removing the sender from the critical path. Many of the time synchronization protocols use a sender to receiver synchronization method where the sender will transmit the timestamp information and the receiver will synchronize. RBS is different because it uses receiver to receiver synchronization. The idea is that a third party will broadcast a beacon to all the receivers (A and B) . The beacon does not contain any timing information; instead the receivers will compare their clocks (ta and tb) to one another to calculate their relative phase offsets. The timing is based on when the node receives the reference beacon.

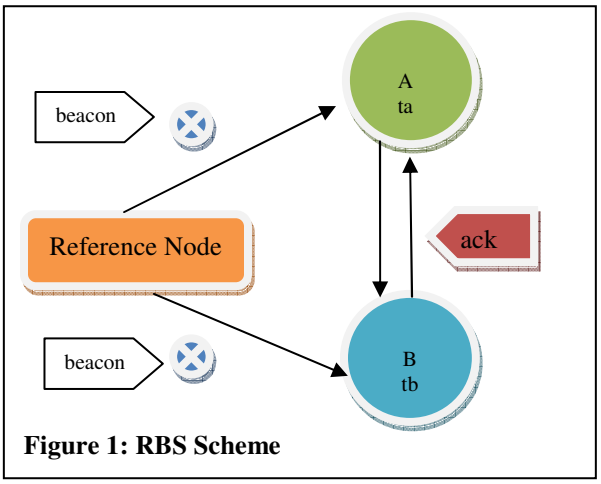

Figure 1: RBS Scheme

The simplest form of RBS is one broadcast beacon and two receivers. The timing packet will be broadcasted to the two receivers. The receivers will record when the packet was received according to their local clocks. Then, the two receivers will exchange their timing information and be able to calculate the offset [3]. This is enough information to retain a local timescale. RBS can be expanded from the simplest form of one broadcast and two receivers to synchronization between n receivers, where n is greater than two. This may require more than one broadcast to be sent. Increasing the broadcasts will increase the precision of the synchronization. The following figure shows the exchange of timing information with their neighboring nodes in RBS scheme, as the nodes will then be able to calculate their offset as seen above in fig 1.

Reference Broadcast scheme (RBS) eliminate the uncertainty of the sender by removing the sender from the critical path. By removing the sender, the only uncertainty is the propagation and receives time. The propagation time is negligible in networks where the range is relatively small. It is claimed that the reference beacon will arrive at all the receiving nodes instantaneously. By removing the sender and propagation uncertainty the only room for error is the receiver uncertainty. TPSN is a traditional *sender-receiver* based synchronization that uses a tree to organize the network topology [4] [5] [6]. The concept is broken up into two phases, the level discovery phase and the synchronization phase. The level discovery phase creates the hierarchical topology of the network in which each node is assigned a level. Only one node resides on level zero, the root node. In the synchronization phase all *i* level nodes will synchronize with *i-1* level nodes. This will synchronize all nodes with the root node. The level discovery phase is run on network deployment. First, the root node should be assigned. If one node was equipped with a GPS receiver, then that could be the root node and all nodes on the network would be synced to the world time. If not, then any node can be the root node and other nodes can periodically take over the functionality of the root node to share the responsibility [3][4][5].

Once the root node is determined, it will initiate the level discovery. The root, level zero, node will send out the *level_discovery* packet to its neighboring nodes. Included in the *level_discovery* packet is the identity and level of the sending node. The neighbors of the root node will then assign themselves as level one. They will in turn send out the *level_discovery* packet to their neighboring nodes. This process will continue until all nodes have received the *level_discovery* packet and are assign a level. Once again all nodes are assigned a level to create a tree type topology. The root node is level zero continuing down the tree with level one and so on. All nodes of level *i* will broadcast the *level_discovery* with all nodes of level *i-1*. This is maintained until all nodes are assigned a level [5].

The basic concept of the synchronization phase is two-way communications between two nodes. As mentioned before this is a sender to receiver communication. Similar to the level discovery phase, the synchronization phase begins at the root node and propagates through the network.

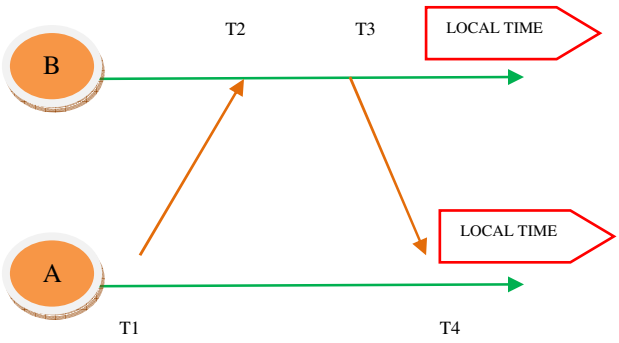

Figure 2: Two –way messaging

Figure 2 illustrates the two-way messaging between a pair of nodes. This messaging can synchronize a pair of nodes by following this method. The times T1, T2, T3, and T4 are all measured times. Node A will send the *synchronization_pulse* packet at time T1 to Node B. This packet will contain Node A's level and the time T1 when it

was sent. Node B will receive the packet at time T2. Time T3 is when Node B sends the *acknowledgment_packet* to Node A. That packet will contain the level number of Node B as well as times T1, T2, and T3. By knowing the drift, Node A can correct its clock and successfully synchronize to Node B. This is the basic communication for TPSN. The synchronization process is again initiated by the root node. It broadcasts a *time_sync* packet to the level one nodes. These nodes will wait a random amount of time before initiating the two-way messaging. The root node will send the acknowledgment and the level one nodes will adjust their clocks to be synchronized with the root nodes.

The level two node will be able to hear the level one nodes communication since at least one level one node is a neighbor of a level two node. On hearing this communication the level two nodes will wait a random period of time before initiating the two-way messaging with the level one nodes. This process will continue until all nodes are synchronized to the root node. Again the synchronization process executes much the same as the level discovery phase. All communication begins with the root node broadcasting information to the level 1 nodes. This communication propagates through the tree until all level *i-1* nodes are synchronized with the level *i* nodes. At this point all nodes will be synchronized with the root node [4] [5].

3.1. UNCERTAINITY IN THE SYNC PACKET

Any synchronization packet has the four delays discussed in [7] send time, access time, propagation time, and receive time. Eliminating any of these would be a plus. A hardware RF transreceiver implementation for reception time is discussed in detail in [7]. Although TPSN does not eliminate the uncertainty of the sender it does, however, minimize it. Also, TPSN is designed to be a multi-hop protocol; so transmission range is not an issue. Unlike RBS, TPSN has uncertainty in the sender. They attempt to reduce this non-determinism by time stamping packets in the MAC layer. It is claimed that the sender's uncertainty contributes very little to the total synchronization error. By reducing the uncertainty with low level time stamping, it is claimed that TPSN has a 2 to 1 better precision than RBS and that the sender to receiver synchronization is superior to the receiver to receiver synchronization. [2]. RBS also is limited by the transmission range. It was stated that RBS can ignore the propagation time if the range of transmission was relatively small. If it is a large multi-hop network, this is not the case. RBS would have to send more reference beacons for the node to synchronize. TPSN on the other hand was designed for multi-hop networks. Their protocol uses the tree based scheme so the timing information can accurately propagate through the network.

Lightweight Tree-based Synchronization (LTS), proposed by Greunen and Rabaey [8] is distinguished from other work in the sense that the aim is not to maximize accuracy, but to minimize the complexity of the synchronization. Thus the needed synchronization accuracy is assumed to be given as a constraint, and the target is to devise a synchronization algorithm with minimal complexity to achieve given precision. This approach is supported by the claim of authors that the maximum time accuracy needed in sensor networks is relatively low (within fractions of a second), and therefore it is sufficient to use a relaxed, or lightweight, synchronization scheme in sensor networks. The algorithm is a centralized algorithm, and needs a spanning tree to be constructed first. Then pair wise synchronization is done along the n - 1 edges of the spanning tree. In the centralized algorithm, the reference node is the root of the spanning tree and has the responsibility of initiating a resynchronization" as needed. Using the assumption that the clock drifts are bounded, and given the required precision, the reference node calculates the time period that a single synchronization step will be valid. Since the depth of the spanning tree affects the time to synchronize the whole network, and also the precision error at the leaf nodes, the depth of the tree is communicated back to the root node so that it can use this information in its resynchronization time decision [8].

Another time-sync class of protocol is Flooded Time Sync protocol (FTSP) [9]. This protocol is similar to TPSN, but it improves on the disadvantages to TPSN. It is similar in the fact that it has a structure with a root node and that all nodes are synchronized to the root. The root node will transmit the time synchronization information with a single radio message to all participating receivers. A Radio Channel cannot be accessed simultaneously by two or more nodes that are in a radio interference range—neighboring nodes may cause “conflict” or signal interference at some nodes if transmitting at the same time on the same channel. In wireless sensor networks, controlling access to the channel, generally known as *medium access control* (MAC), plays a key role in determining channel utilization, network delays, and, more important, power consumption, also influencing congestion and fairness in channel usage.

The message contains the sender's time stamp of the global time at transmission. The receiver notes its local time when the message is received. Having both the sender's transmission time and the reception time, the receiver can estimate the clock offset. The message is MAC layer time stamped, as in TPSN, on both the sending and receiving side. To keep high precision compensation for clock drift is needed. FTSP uses linear regression for this. FTSP was designed for large multi-hop networks. The root is elected dynamically and periodically reelected and is responsible for keeping the global time of the network. The receiving nodes will synchronize themselves to the root node and will organize in an ad hoc fashion to communicate the timing information amongst all nodes. The network structure is mesh type topology instead of a tree topology as in TPSN. Typical WSN operate in areas larger than the broadcast range of a single node; therefore, the FTSP provides multi-hop synchronization. The root of the network—a single, dynamically (re) elected node—maintains the global time and all other nodes synchronize their clocks to that of the root. The nodes form an adhoc structure to transfer the global time from the root to all the nodes, as opposed to a fixed spanning-tree based approach. This saves the initial phase of establishing the tree and is more robust against node and link failures and dynamic topology changes.

3.2. SECURE TIME SYNCHRONIZATION

The primary functionality of wireless sensor networks is to sense the environment and transmit the acquired information to base stations for further processing with secure time information. Studies and experiences (e.g., [10-14, 22-24]) have shown that considering security in the design stage is the best way to provide security for sensor network routing. Several recent contributions to the literature have addressed security and privacy issues in sensor networks [9-13,16] for routing, but it is difficult to implement along with existing time synchronization approaches because they require lots of computations for routing. Thus, the existing time synchronization schemes in sensor networks were not designed with security in mind, thus leaving them vulnerable to security attacks in addition to establishing the need of secure time sync for individual sensor nodes. Security in RBS Scheme is vulnerable and it is prone to many attacks [11]. Taking the RBS scheme as an example, an attacker may launch different kinds of attacks to break the protocol.

In [11], the authors discuss four different type of attacks. The first attack is called *masquerade attack*. Suppose a node A sends out a reference beacon to its two neighbors B and C. An attacker E can pretend to be B and exchange wrong time information with C, disrupting the time synchronization process between B and C. A second attack is called *replay attack*. Using the same scenario in the first attack, the attacker E can replay B's old timing packets, misleading C to be synchronized to a wrong time. A third attack is called *message manipulation attack*. In this attack, an attacker may drop, modify, or even forge the exchanged timing messages to interrupt the time synchronization process.

For the message dropping attack, the attacker can selectively drop the packets and thus prolong the converging time of the synchronization process. This can be done on a random or arbitrary basis, making it more difficult to be detected. For the message forging attack, the attacker can forge many reference beacon messages and flood the network. This not only incorrectly synchronizes the neighbors, but also causes those nodes to consume power to process these unwanted and faked timing messages. If some nodes run out of power, coverage holes or network partitions may appear. The last type of attack is the Delay attack, where, the attacker deliberately delays some of the time messages (e.g., the beacon message in the RBS scheme) so as to fail the time synchronization process. It is noted that this attack cannot be defended against by cryptographic techniques. We can certainly employ some cryptographic techniques [12][13] to address the aforementioned attacks. For example, providing authentication of every exchanged message will prevent an outside attacker from impersonating other nodes or altering the content of an exchanged message. Adding a sequence number to beacon messages or other messages will prevent message replay attacks.

For RBS, an attack on the synchronization can be executed easily. RBS works by receiver to receiver synchronization in which both nodes receive the reference beacon and then calculate their offset with one another. An attack would be as simple as compromising one of the nodes with an incorrect time. The non compromised node will then calculate an incorrect offset during the exchange period. Remember TPSN is a sender to receiver tree based protocol with two phases, the level discovery phase and the synchronization phase. Both of the phases are initiated by the root node. In the synchronization phase the level number and the time are both sent through the tree.

An attack would simply be to compromise a non-root node with the incorrect time. This will propagate through the tree and the closer the compromised node is to the root node, the more incorrect synchronization will occur. Also a node could lie about its level. That would cause other nodes to request synchronization information in which it could give inaccurate information. That node also could refuse to participate in the level discovery phase, which could eliminate its children from the network. The fundamental problem in FTSP [9] is that it allows for any node to elect itself the root after a period of time of not receiving the synchronization information. A corrupt node could claim itself to be the root and now the other nodes will respond to its timing information instead of the correct information from the real root node. The will of course propagate through the network until all nodes have incorrectly calculated their skew and offset. Since none of the protocols were designed with security in mind, attacks on the synchronization are easily executed by following the rules of the protocol. In the sender to receiver synchronization, an attack will institute more damage because it will propagate through the network. Several papers have studies on security issues [8-13] in sensor network which are attack resilient.

We now discuss another class of sync protocol called the Gradient Time Synchronization Protocol (GTSP) which is a completely distributed time synchronization protocol [14]. In this sync method, nodes periodically broadcast synchronization beacons to their neighbors. Using a simple update algorithm, they try to agree on a common logical clock with their neighbors. It can be shown by theoretical analysis that by employing this algorithm, the logical clock of nodes converges to a common logical clock. GTSP relies on local information only, making it robust to node failures and changes in the network topology. Experiments on a test bed setup of 20 Mica2 nodes and simulations showed that the remaining synchronization error between neighbors is small while still maintaining an acceptable global skew. Furthermore, the authors have shown that GTSP can improve the synchronization error between neighboring sensor nodes compared to tree-based time synchronization protocols. The FTSP clock synchronization algorithm exhibits an error that grows exponentially with the size of the network, for instance [9]. Since the involved parameters are small, the error only becomes visible in midsize networks of about 10-20 nodes. In contrast, the authors of another class of sync protocol *PulseSync*, discussed a new clock synchronization algorithm that is asymptotically optimal. They evaluate *PulseSync* on a Mica2 testbed, and by simulation on larger networks. On a 20 node network, the prototype implementation of *PulseSync* outperforms FTSP by a factor of 5. Theory and simulation show that for larger networks, PulseSync offers an accuracy which is several orders of magnitude better than FTSP [9].

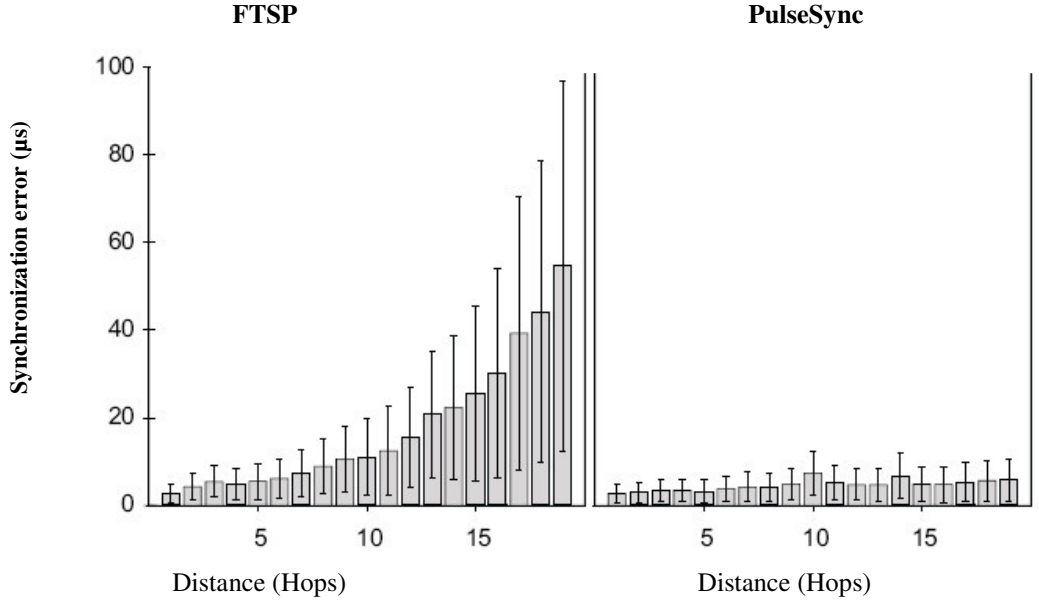

Figure 3: Synchronization error in (µS) vs number of Hops for FTSP and PulseSync

To round off the presentation, the *PulseSync*, investigate several optimization issues,e.g. media access and local skew[15].Furthermore, the authors of *PulseSync* claim that *PulseSync* converges rapidly to a common logical time since the time information is flooded in a single round from the reference node to all other nodes throughout the network. The authors proved that the *PulseSync* and FTSP have the same message complexity (one message per node and synchronization period), the fast flooding approach of Pulse-Sync achieves significantly improved synchronization accuracy on the same network topology. Furthermore, their experiments confirm with a finding that the synchronization error to root node must increase exponentially when using FTSP, while PulseSync performs better as shown in above figure[15].

3.3. ENERGY CONSUMPTION OF SYNCHRONIZATION SCHEMES

Most wireless sensor networks to date employ nodes with limited power, processing, storage, and communication capabilities. Additionally, they typically include relatively simple sensors (e.g., light, temperature, pressure, magnetometer, etc.). In these deployments, energy consumed by sensing-related tasks is relatively low, which means that the communication subsystem (i.e., the radio) dominates energy consumption. RBS and TPSN both achieve accurate clock synchronization within a few microseconds of uncertainty. However, they are both designed for networks with a small number of sensors and are not specifically geared towards energy conservation; although these algorithms will work for larger networks, their energy consumption becomes inefficient and network connectivity is not maintained once nodes begin losing power. Simulating each of these methods shows that synchronizing a large sensor network requires an unnecessarily large number of transmissions, which will quickly deplete sensors and reduce the network's coverage area.

The hybrid algorithm proposed in [7] chooses RBS over TPSN based on receiver threshold and number of receivers. The results from Table 3.1. show that RBS's energy consumption is more dependent on the density of sensors in a given area. In contrast, TPSN and the hybrid algorithm are less affected by the size of the network. When the network size increases from 250 sensors to 500 sensors (for the same area of 1 km^2), RBS becomes less energy efficient than TPSN. The hybrid algorithm outperforms TPSN by 15.7%, while outperforming RBS by 20.8%. Once the network increases to 750 sensors, RBS clearly becomes less efficient than TPSN. The hybrid algorithm still outperforms TPSN by 12.7%. Since RBS consumes more energy, the hybrid algorithm now outperforms it by 32%. Hybrid algorithm, the power reduction is even more drastic in large multi-hop sensor networks. The method also improves upon these algorithms by maintaining a large area of coverage even when some sensors lose power [7][17]. Please see Table 3.1 for the savings on energy in milliWatts range with no of sensors used[]. For the MICA2Dot platform [18], a reception uses approximately 24 mW of power, while a transmission requires 75 mW at -5 dBm.

TABLE 3.1: AVERAGE ENERGY CONSUMPTION (mW) IN RBS, TPSN & HYBRID SYNC METHODS

# Sensors	250	500	750	1000	1250	1500
RBS	466	1046	1844	2762	3756	5060
TPSN	511	983	1434	1885	2331	2770
Hybrid	404	828	1253	1672	2095	2514
Savings over RBS	9.29%	20.8%	32.0%	39.4%	44.2%	50.3%
Savings over TPSN	20.8%	15.7%	12.7%	11.2%	10.1%	9.2%

3.4. POST-FACTO SYNCHRONIZATION

To save energy in a sensor network, it is a desirable to keep nodes in a low-power state, if not turned off completely, for as long as possible. Sensor network hardware is often designed with this goal in mind; processor have various "sleep" modes or are capable of powering down high-energy peripherals when not in use. In post-facto scheme proposed by Jeremy Elson and Deborah Estrin [3], nodes' clocks are normally *un*synchronized. When a stimulus (signal) arrives, each node records the time of the stimulus with respect to its own local clock. Immediately afterwards, a "third party" node acting as a beacon broadcasts a synchronization pulse to all nodes in the area using its radio. Nodes that receive this pulse use it as an instantaneous time reference and can normalize their stimulus timestamps with respect to that reference. This kind of synchronization is not applicable in all situations, it is limited in scope to the transmit range of the beacon and creates only an "instant" of synchronized time as discussed in the paper [3].

This makes it inappropriate for an application that needs to communicate a timestamp over long distances or times. However, it does provide exactly the service necessary for beam-forming applications, localization systems, and other situations in which we need to compare the relative arrival times of a signal at a set of spatially local detectors. Post-facto sync method need to be tested in real WSN.

3.5. GLOBAL CLOCK SYNCHRONIZATION

Many emerging sensor network applications require that the sensors in the network agree on the time. A global clock in a sensor system will help process and analyze the data correctly and predict future system behavior. For example, in the vehicle tracking application, each sensor may know the time when a vehicle is approaching. By matching the sensor location and sensing time, the sensor system may predict the vehicle moving direction and speed. Without a global agreement on time, the data from different sensors cannot be matched up. There are several algorithms developed and studies in literatures [19] [20] using global sync messages such as the basic and popular all-node approach by Qun Li and Rus [19] shown in Figure 4. The main idea of their algorithm is to send a message along a loop and record the initial time and the end time of the message. Then by using the message traveling time, we can average the time to different segments of the loop and smooth over the error of the clocks. But the all node method are not scalable for very large networks. The initiating node may encounter failure and thus the approach is not fault tolerant. The nodes that participate in the synchronization must execute the related code approximately at the same time, which may be too hard in a large system.

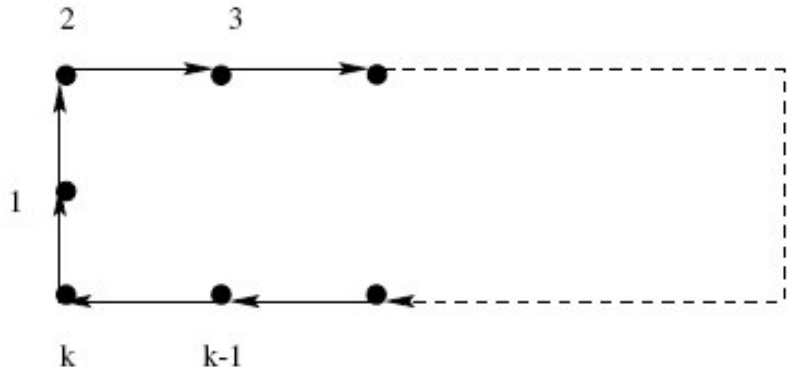

Figure 4: Qun Li's - All-node method

There are also approaches which uses diffusion techniques [21] either they diffuse low or high or average clock readings from one node to next or neighboring node. The average diffusion based algorithm is more robust and reasonable, Since they use global average value as the ultimate synchronization clock reading. The main idea of the

algorithms is to average all the clock time readings and set each clock to the average time. A node with high clock time reading diffuses that time value to its neighbors and levels down its clock time. A node with low time reading absorbs some of the values from its neighbors and increases its value. After a certain number of rounds of diffusion the clock in each sensor will have the same value, but these methods use global time information sent to all the nodes and thus they are not scalable for very large networks. The initiating node may encounter failure and thus the approach is not fault tolerant. The nodes that participate in the Synchronization must execute the related code approximately at the same time, which may be too hard in a large system. Diffusion techniques may avoid this scalability problem, where synchronization is done locally by spreading the local synchronization information to the whole system. It can choose various global values to synchronize the network provided that each node in the overall network agrees to change its clock reading to the consensus value. An easy and possible way is to choose the highest or lowest reading over the network. Several paper contributions were made on diffusion techniques [21].

A master-slave protocol assigns one node as the master and the other nodes as slaves. The slave nodes consider the local clock reading of the master as the reference time and attempt to synchronize with the master. In general, the master node requires CPU resources proportional to the number of slaves, and nodes with powerful processors or lighter loads are assigned to be the master node. Mock et al. [25] have adopted the IEEE 802.11 clock synchronization protocol due to its simple, non-redundant, master/slave structure. Ping's protocol [26] also adheres to the master-slave mode.

Nazemi et.al [27] used passive clustering and linear regression Scalable Light weight Time Sync protocol (SLTP) to reduce the energy consumption of network nodes and also decrease the overhead of creating and maintaining the clusters. Moreover SLTP uses linear regression to compute the time. Therefore, it can calculate the clock skew and offset between each node and its cluster head in order to estimate the local time of remote nodes in the future or the past. Simulation results using NS2 Simulator show that the SLTP can gain considerable improvements in power consumption, accuracy and scalability in comparison to similar algorithms [27][15]

TABLE 3.2: A COMPARISON CHART OF VARIOUS TIME SYNC SCHEMES AND METRICS

Time Sync methods	Accuracy	Energy Efficiency	Complexity	Scalability	Fault Tolerance	Byte Alignment
1. RBS	Low, 29.1µs is the average error per hop	High	High	Good	No	Note Handled
2. TPSN	16.9µs per hop	High	Low	Poor	No	Not Handled
3. FTSP	1.48 µs per hop	High	High	Average	No	Handled
4. GTSP	Average network wide error, 14µs, Avg. neighbor sync error is 4µs	High	Average	Good	Yes	Not Handled
5. PulseSync	4.4µs (compared to 23.96µs for FTSP between any two nodes)	High	High	Good	No	Unknown
6.All-node	Low	Low	Low	No	No	No
7.Diffusion techniques	Average	Average	High	Good	No	No
8. Mock et.al	High	Low	Low	NA	Yes	Unknown
9 .Ganeriwal [4, 5]	High	Average	Low	Good	Yes	No
10 .Ping	High	High	Low	Good	Yes	Unknown
11.SLTP	High	High	Average	High	No	NA

4. GROUP SYNCHRONIZATION

All current approaches use the technique of pair wise synchronization but this is insufficient in cases where a group of nodes need to be synchronized for instance in tracking application. In a group sync model, a single beacon beacons, all receivers respond with their offset to the beacon. The beacon has one measurement to relate each pair of nodes which either form a sender-receiver (sender-receiver) pair or a receiver-receiver (receiver-receiver) pair. This approach will have a 4 times extra cost but will result in only one measurement between any pair of sensor nodes resulting in achieving better accuracy readings. One way to do this through beam-forming arrays, which can perform "spatial filtering", receiving only signals arriving from a certain direction. This depends on the relative time offsets of the array's sensors. Another problem in Synchronization algorithm is duplicate detection of events. The time of an event helps nodes in the cluster determine if they are seeing two distinct real-world events, or a single event seen from two vantage points. If they are indeed seeing the same event, they can further fuse their observations to get much meaningful information about the event. All these applications will function accurately only if the synchronization error between nodes in a group is bounded. So far, all the existing work in the research literatures concentrates on establishing pair wise relationships between a pair of nodes at a given instant of time. However, there is need of developing a unified framework that uses one optical beacon to synchronize all nodes in group or network wide at the instant of time[16][15][17].

5. CONCLUSION

A family of existing time synchronization algorithms are studied and a comparative summary has been presented in the paper. As increasing demand in use and promising application of sensor network are emerging, the need of very precise secure clock measurement algorithms are vital for error free clock time measurements whether it is loosely or densely packed or it is deployed in star or mesh or any other framework. The authors strongly believe that using this comparative survey paper, it will make future researchers to explore existing time sync approaches with much easier and will give them a choice to implement their application based on our study on various time sync protocols.

6. ACKNOWLEDGEMENTS

The authors are thankful to all synchronization papers that are credited as references below. This is a comprehensive study paper on synchronization algorithms or approaches collectively in one paper form cited from various contributions from the authors mentioned below.

REFERENCES

[1] S. Ganeriwal, M. Srivastava, "Timing-sync Protocol for Sensor Networks (TPSN) on Berkeley Motes," *NESL*, 2003.

[2] NTP: The Network Time Protocol, http://www.ntp.org/.

[3] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *ACM OSDI 2002*,Boston, MA, December 2002.

[4] Kay Romer. Time synchronization in ad hoc networks. In *ACM MobiHoc*, Long Beach, CA, Oct. 2001.

[5] S. Ganeriwal, R. Kumar, M. Srivastava. "Timing Sync Protocol for Sensor Networks," *ACM SenSys '03*, 2003.

[6] Chipcon CC1000 Radio Datasheet,http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf

[7] T Robert Akl Yanos Saravanos, The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07) Hybrid Energy – aware Synchronization algorithm in Wireless sensor networks, 2007.

[8]. J.V. Greunen, and J. Rabaey, Lightweight Time Synchronization for Sensor Networks", *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications(WSNA)*, San Diego, CA, September 2003.

[9] Maroti, M., Kusy, B., Simon, G., and Ledeczi, A., The Flooding Time Synchronization Protocol, *Proc. of the 2nd ACN Conf. on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, 2004, pp. 39–49

[10] C. Karlof and D. Wagner, "Secure Routing in Sensor Networks:Attacks and Countermeasures," *Proc. 1st IEEE Int'l.Wksp. Sensor Network Protocols and Apps.*, 2003.

[11] H. Chan and A. Perrig, "Security and Privacy in Sensor Networks," *Computer*, vol. 36, no. 10, Oct. 2003, pp. 103–05.

[12] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution
Schemes for Sensor Networks," *Proc. 2003 IEEE Symp. Sec. and Privacy*, May 2003, pp. 197–213.

[13] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Comp. and Commun. Sec.*, 2003, pp. 52–61.

[14] Philipp Sommer, Roger, Wattenhofer, Gradient clock synchronization in wireless sensor networks**,** *Proceedings of the International Conference on Information Processing in Sensor Networks,*2009

[15] Optimal Clock Synchronization in Networks, Christoph Lenzen et al., Sensys 2009.

[16] Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, Mani B. Srivastava , Secure Time Synchronization Service for Sensor Networks, *WiSE'05*, September 2, 2005, Cologne, Germany

[17] G. Asada, M. Dong, T. Lin, F. Newberg, G. Pottie, W. Kaiser, and H. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In *Proceedings of the European Solid State Circuits Conference*, 1998.

[18] Crossbow MICA2Dot Wireless Microsensor Mote, Document Part Number 6020-0043-05 Rev A, http://www.xbow.com/Products/ Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf.

[19] Qun Li*, Daniela Rus* "Global Clock Synchronization in Sensor Networks*,* IEEE INFOCOM 2004

[20] Cheng Liao, Margaret Martonosi, and Douglas W. Clark. Experience with an adaptive globally-synchronizing clock algorithm. In *ACM SPAA*, pages 106–114, New York, June 1999.

[21] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MobiCOM 2000*, Boston, Massachusetts, August 2000.

[22] H. Song, S. Zhu, and G. Cao, "Attack-Resilient Time Synchronization for Wireless Sensor Networks," *Proc. 2nd IEEE Int'l. Conf. Mobile Ad Hoc and Sensor Sys.*,Washington, DC, Nov. 2005.

[23] N. Gura *et al.*, "Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs," *Proc. 6th Int'l. Wksp. Cryptographic Hardware and Embedded Sys.*, Boston, MA, Aug. 2004.

[24] D. Malan, M. Welsh, and M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography," *Proc. 1st IEEE Int'l. Conf. Commun. and Networks*, Santa Clara, CA, Oct. 2004.

[25] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous Clock Synchronization in Wireless Real-time Applications.*Proc. 19th IEEE Symposium on Reliable Distributed Systems (SRDS-00),* pp. 125–133, Oct. 2000.

[26]S. Ping. Delay Measurement Time Synchronization for Wireless Sensor Networks. *Intel Research, IRB-TR-03-013,* June 2003.

[27] SLTP: Scalable Lightweight Time Synchronization Protocol for Wireless Sensor Network *Sepideh Nazemi Gelyan, Arash Nasiri Eghbali, Laleh Roustapoor, Seyed Amir Yahyavi Firouz Abadi, and Mehdi Dehghan,* Springer-Verlag Berlin Heidelberg 2007.

Authors

Prakash Ranganathan is a faculty in Electrical Engineering at University of North Dakota, Grand Forks. He earned his undergraduate degree from University of Madras, India and graduate degree from North Dakota State University, Fargo. He is currently teaching Electric Circuits, Computer Aided Measurements and controls. His research interests are in Sensor Networks, Smart Grid technologies and Engineering Education. He is active member of IEEE and ASEE.

Kendall E. Nygard is Professor of Computer Science and Operations Research at North Dakota State University where he has served on the faculty since 1977. From 1996 to 2005 he served as Department Chair. In 1986 and 1987 he was on the adjunct faculty in operations research at the Naval Postgraduate School in Monterey, California. In 1994 and 1995 he was Director of the Scientific Computing Center at the University of North Dakota in Grand Forks. In 1984 he was a visiting scientist at the Air Force Logistics Command at Wright-Patterson AFB in Ohio. In 2000 he was a research fellow at the Air Vehicle Directorate of the Air Force Research Lab. His research areas involve combinatorial optimization methods, with applications to management of networks and sensor networks, cooperative mission control for unmanned air vehicles (UAVs), computer-based transportation analysis, and bioinformatics.