International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 2 Issue 5, May-2013, pp: (61-67), Available online at: www.erpublications.com

# Time Synchronization in WSN: A survey

Vikram Singh, Satyendra Sharma, Dr. T. P. Sharma
NIT Hamirpur, India

**Abstract: Wireless sensor networks have emerged as an important and promising research area in the recent years. They are a special type of ad-hoc networks, where wireless devices collaborate with other devices to send data to the destination. The nodes collect the sensed data, process them, and transmit that data over the communication channel, which is broadcast by nature. Synchronization is an important issue for wireless sensor networks because temporal coordination is required for many of the collaborative tasks they perform. E.g. For the task of Data Fusion, in object tracking and velocity estimation, in setting the sleep modes of the various nodes so that the battery life is prolonged, etc.. There are several synchronization schemes which have been put forward till date. In this paper, the schemes existing presently have been described.**

## I. INTRODUCTION

The advancement in the technology has enabled the development of tiny devices which are capable of sensing, processing, and communicating with each other. Wireless Sensor Networks are a special type of ad-hoc networks, where tiny wireless devices collaborate with other devices to send data to the destination in a multi-hop communication environment. These devices or nodes have their own characteristics such as energy constraints, inexpensive, small in size, unreliable, etc. These nodes collect the sensed data, process them, and transmit that data over the communication channel, which is broadcast by nature. Synchronization is an important issue for wireless sensor networks because temporal coordination is required for many of the collaborative tasks they perform. E.g. For the task of Data Fusion, in object tracking and velocity estimation, in setting the sleep modes of the various nodes so that the battery life is prolonged, etc..

There are several requirements that determine what kind of synchronization technique to use. Some of them are:

- **Energy efficiency:** The synchronization technique should take into consideration the limited energy resources available. Energy is the most important factor because the sensors work on batteries and the replacement of the batteries is difficult, or even impossible in many situations.
- **Scalability:** Synchronization scheme should also scale well with the network size. As the sensor nodes increases/decreases the synchronization scheme should be able to sustain the change in topology.
- **Precision:** Some applications might need microsecond accuracy while others may just require the ordering of the events. So, according to the requirement, the appropriate scheme can be used.
- **Robustness:** Synchronization scheme should be robust against the link and node failures. For this purpose, several sensor nodes are deployed in a relatively smaller area as compared to other networks.
- **Lifetime:** Lifetime decides whether the synchronization is needed for an instant, or for the entire lifetime of the network. Synchronization for the longer period of time might require regular synchronization.
- **Scope:** The scope decides whether the scheme provides the network wide External Synchronization or only local synchronization among the nodes which are spatially close.
- **Cost:** How much cost is incurred while deploying the scheme. The cost should not exceed too much because of the reason that the sensors may be deployed in harsh conditions, and thus may be prone to failures. Since the sensor networks are often deployed in remote areas, it is better not to rely on the sophisticated hardware infrastructures like GPS receivers. Rather, an internal synchronization is enough if implemented appropriately.

But, there are some challenges to the wireless synchronization such as nondeterministic delays. When a node sends a timestamp value to another node, the packet may face a variable amount of delay before it reaches the receiver. This delay might prevent the two nodes to be synchronized accurately. Some of the sources of errors are:

- **Send Time:** It consists of the time spent in constructing the message and transferring to the network interface. Also includes the kernel processing, operating system overheads like context switch, etc.
- **Access Time:** It is the time taken to access the transmit channel. The reasons for the delay could be waiting for the channel to be idle, waiting for its slot to transmit, etc. This depends on the type of MAC scheme used.
- **Transmission/Reception Time:** Time taken to send/receive the message bit-by-bit at the physical layer. Depends on

the packet size and the baud rate of the transmission.

- **Propagation Time:** Time taken to propagate the message from the sender to the receiver. This is small enough in most cases to be ignored from latency estimations.
- **Receive Time:** Time taken to notify the host about the reception of the message by the network interface.

## II. SYNCHRONIZATION TYPES

By time synchronization, we mean that each and every clock in the network shows the same time. There are two approaches to achieve this goal. One is the External Synchronization and the other is the Internal Synchronization. In the External Synchronization, the nodes get the time from some external source of the standard time. This type of synchronization is also known as Global Synchronization. But achieving such kind of synchronization can be expensive because it will require that some sophisticated devices, like GPS receivers, be attached to the sensor nodes. And this requirement can lead to a very expensive deployment of the network. So, another type of synchronization can be employed, which is the Internal Synchronization. In this, the nodes agree on a particular time for a network, although they may not agree to the global time. This method is much economical and doesn't even require costly devices attached to the nodes.

## III. CLOCK MODEL

Time Synchronization aims at providing a common time scale for the clocks of the nodes in the network. But the clocks are not perfect and so the keep on drifting away from each other over the time. Therefore the clocks that have even been synchronized once may show different time after some time.

In software, a clock $C(t)$ is described as:

$$C(t) = \alpha t + \beta$$

where, $\alpha$ is the skew rate of the clock and determines how much time the clock will gain or lose over a given period and $\beta$ is the offset, which determines the variation from the real time. For a perfect clock, the skew rate is 1. In reality, skew rate is not static over time. Therefore a synchronization scheme should equalize the clock rates as well as offsets, and then should repeatedly correct the offsets to keep the clocks synchronized over a time period.

A tolerance value $\rho$ in parts per million (PPM) is also specified by the manufacturers, which determines the maximum amount that the skew rate will deviate from 1.

$$1-\rho \le \alpha \le 1+\rho$$

## IV. EXISTING SCHEMES

### 1. Reference Broadcast Synchronization

This scheme was proposed by Elson, Girod and Estrin in the year 2002. In this scheme, the nodes transmit reference beacons to their neighbors via physical-layer broadcast. This broadcast doesn't contain any timestamp explicitly. In fact the arrival time of the beacon is considered as the point of reference for comparing the clocks. Doing so removes the Send Time and the Access Time from the critical path. In RBS, the critical path length is the time from the injection of the packet into the channel to the last clock read. Thus, this scheme removes the sender's nondeterminism from the critical path as shown in the fig 1. Any extant broadcast can be used to get the timing information. No dedicated timesync packet broadcast is needed.

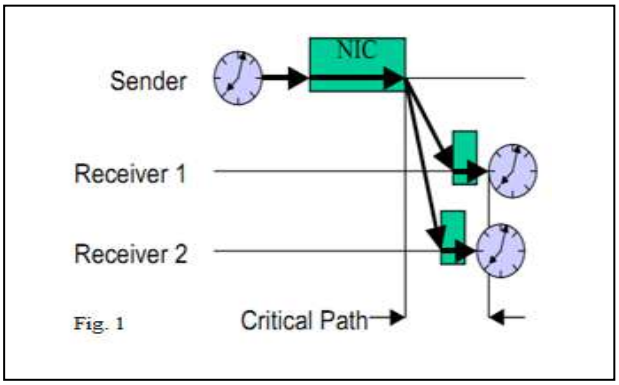

Fig. 1

**Estimation of Phase Offset:** The simplest form of RBS is the broadcast of a single pulse to two receivers, allowing them to estimate their relative phase offsets. That is:

1. A transmitter broadcasts a reference packet to two receivers (i and j).
2. Each receiver records the time that the reference was received, according to its local clock.
3. The receivers exchange their observations.

**Advantages:**

1. The largest sources of nondeterministic latency can be removed from the critical path by using the broadcast channel to synchronize receivers with one another. This results in better precision.
2. Multiple broadcasts allow tighter synchronization.
3. RBS can work using local timestamps. Absolute time reference is not required.

This scheme has some limitations too. This scheme works in a network which has a physical broadcast channel. But it cannot be used in the networks which employ point-to-point links.

## 2. Timing-sync Protocol for Sensor Networks

This scheme was proposed by Ganeriwal et.al. in the year 2003. This protocol aims at providing network-wide synchronization. The difference between RBS and Timing-sync Protocol for Sensor Networks (TPSN) is that TPSN uses the classical approach of sender-receiver synchronization whereas RBS uses the approach of receiver-receiver synchronization. In TPSN, initially a hierarchical topology is created. A level is assigned to each node in that hierarchical structure. The node assigned the level 0 is known as 'root' node. This establishing of the hierarchical structure is called "level discovery phase". The root node then initiates the second stage of the scheme which is known as "synchronization phase". In this phase, a level 'i' node will synchronize to a level 'i-1' node. Eventually all the nodes in the hierarchical structure will be synchronized. The root node can be chosen using some leader election algorithm.

**Level Discovery Phase:** This is the first phase and starts when the network is deployed. The root node is assigned the level 0 and it then initiates this phase by broadcasting a level_discovery packet. This packet contains the identity and the level of the sender. The immediate neighbors receive this packet and assign themselves one level greater than the sender's level. After assigning themselves a level, they too broadcast the level_discovery packet containing their own level. The process continues and each node is assigned a level. Once assigned a level, such packets are neglected to check the congestion.

**Synchronization Phase:** A two-way message exchange is used in this phase. Here, T1, T4 represent the time measured by local clock of 'A'. Similarly T2, T3 represent the time measured by local clock of 'B'. At time T1, 'A' sends a synchronization pulse packet containing its level number and the value of T1 to 'B'. Node 'B' receives the packet at time T2 where T2 is equal to T1 + Δ + d. Here Δ and d represents the clock drift between the two nodes and propagation delay respectively. At time T3, node 'B' sends an 'acknowledgement' packet containing the level of 'B' and the values of T1, T2, and T3 to node 'A'. Node 'A' receives the packet at time T4. Now node 'A' can calculate the clock drift and propagation delay as follow:

$$\Delta = \frac{(T2-T1)-(T4-T3)}{2}; \quad d = \frac{(T2-T1)+(T4-T3)}{2}$$

Calculating the drift, node A can correct its clock accordingly, and can get synchronized to node B. This approach is called sender-initiated approach because the sender synchronized itself to the receiver.

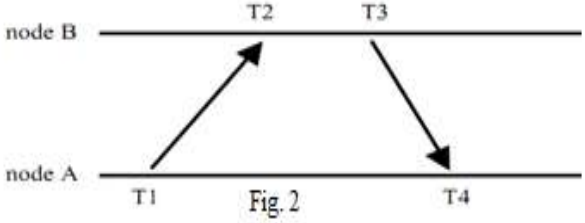

Fig. 2

The message exchange handshake begins after the root node initiates the phase by broadcasting a synchronization packet. Nodes at level 1 on receiving the packet initiates the two-way message exchange. They may take some random time before initiating the two-way message exchange to avoid the contention in medium access. On receiving back an acknowledgement, these nodes adjust their clocks to the root node. On receiving the message exchange, the level 2 nodes back off for some random time and then initiate the message exchange with nodes in level 1. This process continues and eventually all the nodes get synchronized to the root node.
If an elected root node dies, the level '1' nodes won't receive any acknowledgement packet and hence timeout will occur. A leader election is run and a new root is elected. And then the level discovery phase is restarted.

### 3. Lightweight Tree-based Synchronization (LTS)

This scheme was proposed by Greunen and Rabaey. The aim is not to maximize the accuracy, but to minimize the complexity of synchronization. The authors believe that the accuracy needed in sensor networks is relatively low. Two LTS algorithms have been proposed for multihop synchronization of the network. Both these algorithms require the nodes to synchronize to some reference points such as a sink node.

**First algorithm:** It is a centralized algorithm. A spanning tree is constructed first and then the nodes are synchronized along the (n-1) edges of the spanning tree. The root of the spanning tree is the reference node and is responsible for initiating the synchronization. The depth of the spanning tree affects the time to synchronize the whole network. Therefore, the depth is communicated back to the root node so that is can use this information in its resynchronization time decision.

**Second algorithm:** The second algorithm works in a distributed manner. Each node can decide the time for its own synchronization. Spanning tree structure is not used in this algorithm. When a node 'A' needs to be synchronized, it sends a synchronization request to the closest reference node. All the nodes along the path from the node 'A' to the reference node must be synchronized before the node 'A' is synchronized. The advantage of this scheme is that some nodes may not need frequent synchronization. And since in this scheme, the nodes decide of their own synchronization, unnecessary synchronization efforts are saved. Also it may boost the number of pairwise synchronizations, since for each request, all nodes along the path from reference node to the initiator of resynchronization need to be synchronized. The synchronization requests can be aggregated also to reduce the wastage of resources.

### 4. Flooding Time Synchronization Protocol (FTSP)

This scheme was proposed by Maroti et.al in the year 2004. This scheme provides multi-hop synchronization. The root node, which is dynamically elected node, maintains the global time and all other nodes synchronize their clocks to that of the root. An ad-hoc structure is formed by the node to transfer the global time from the root to all the nodes. Spanning tree is not formed in this scheme. This saves the initial phase of establishing the tree and is more robust against node and link failures and dynamic topology changes.

**Multi-hop Synchronization:** Every node in the network has a unique ID. A node broadcasts a synchronization message in order to synchronize other nodes. This synchronization message contains three fields: the timestamp, the rootID, and the seqNum.

The timeStamp contains the global time estimate of the transmitter when the message was broadcasted. The rootID field contains the ID of the root, as known by the sender of the message. The seqNum is a sequence number set and incremented by the root when a new synchronization round is initiated. Each node maintains a highestSeqNum for the purpose of aiding message filtering. Each node has one more variable, known as myRootID. This variable contains the rootID as know by the node. The synchronization message which is then received can be used to create a reference point only if the rootID field of the message is less than or equal to myRootID, and the seqNum field is greater than the highestSeqNum in case the rootID=myRootID.

### 5. Global Clock Synchronization in Sensor Networks

This scheme was proposed by Q. Li et al. in the year 2006. This paper aims at the global synchronization in a sensor network. Four methods have been discussed:

1. The all-node-based method
2. The cluster-based method
3. The fully localized diffusion-based method
4. The fault-tolerant diffusion-based method.

**The all-node-based method:** This method is used on all the nodes in the system and it is most effective when the size of the sensor network is relatively small. The key idea 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, the time in different segments of the loop can be averaged which will smooth over the error of the clocks. In order to synchronize the entire network, paths need to be designed so that they contain all the nodes. The synchronization is divided into two phases. In the first phase, a synchronization packet is sent along a ring. The initiating node of the packet records its local starting time and the ending time of the packet. Each other node simply forwards the packet and records how many hops the packet had traveled thus far. In the second phase, a clock correction packet is sent along the same ring. This packet informs each node of the packet starting and ending time for the initiating node and the total hops in the cycle. Each node then computes its clock adjustment.

**The cluster-based synchronization:** In this, same method is used to synchronize all the cluster heads by designing a message path that contains all the cluster heads. Then, in the second step, the nodes in each cluster can be synchronized with their head.

**The diffusion method:** The main idea here 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 the diffusion, the clock in the network will have the same value.

**The fault-tolerant method:** In this some nodes are considered as tamper-proof (called N nodes), and other normal nodes are called M nodes. A tamper-proof node will destroy itself once it is compromised. This guarantees that an N node can always be trusted. Four basic operations are there:

1. Neighbor discovery: Neighbor discovery for an N node means finding all the neighbors that are shared with another N node.
2. Beacon broadcast: In beacon broadcast operation, an N node 'A' broadcasts a synchronization message to all its neighbors so that each of its neighbors will record the current clock reading.
3. The collect operation: The collect operation is a composite process in which all the neighbors that receive the broadcast send 'A' their clock readings.
4. Broadcast of the average value: In the last step, 'A' broadcasts the average value to all the neighboring nodes and all good neighboring nodes will have a new value after they authenticate the message.

## 6. Fault-Tolerant FTSP Protocol for Wireless Sensor Networks

This protocol, proposed by L. Gheorghe et al. in the year 2010 is a modification of the Flooding Time Synchronization Protocol so that it provides the synchronization even in the presence of malicious nodes. It includes three steps: Fault detection, asking for help, and receiving help and decision.

**Fault Detection:** In this process the node becomes aware that it has received an inconsistent clock value, which might be very different than the other previously received values. This value is represented by having a constant value called the TIME_ERROR_LIMIT that is the maximum difference between sequential received global times.

**Asking for help:** This is the second step of this protocol. When a sensor node receives a reference point that is not compatible with its previous time estimates, it stores the local time and sends a broadcast message that contains a request for the latest global time received from a synchronized node. The neighbors reply with the global time from the last stored reference point.

**Receiving help and decision:** This step decides whether the value was actually faulty or not. If the value was not faulty, the regression table is erased and the new value is stored. Otherwise, a new computed reference point will be inserted into the regression table.

The synchronization message here includes the following fields: timestamp, rootID, seqNum, type and frontierID. The type for normal synchronization message is 0. The synchronization root sends synchronization messages with a frontierID of 1, meaning that it is transmitting to nodes in the first frontier. The nodes in the first frontier send synchronization messages with a frontierID of 2, and so on. The nodes on a certain frontier store the frontierID received in a variable called my FrontierID and they send synchronization messages with the value of (myFrontierID+1).

### 7. Consensus Clock Synchronization for Wireless Sensor Networks

This paper was proposed by M. K. Maggs et al. in the year 2012. Instead of trying to synchronize to an external reference like absolute time t or UTC, the CCS protocol aims to achieve an internal consensus within the network on what time is, and how fast it travels. With each synchronization round the CCS algorithm updates the compensation parameters for each node and over time the network clocks converge to a consensus.

$$\lim_{t\to\infty} \hat{C}_i(t) = C_c(t)$$

This consensus clock is not a physical clock. It is a virtual clock that is generated from the network of nodes running the CSS algorithm. This clock has its own skew rate and offset relative to the absolute time.
By expanding the clock functions from both sides we get the compensation parameters that all nodes must obtain in order to synchronize to the consensus clock.

$$\lim_{t\to\infty} \frac{\alpha_i}{\hat{\alpha}_i(t)} t + \beta_i - \hat{\beta}_i(t) = \alpha_c t + \beta_c$$

$$\lim_{t\to\infty} \hat{\alpha}_i(t) = \frac{\alpha_i}{\alpha_c}$$

$$\lim_{t\to\infty} \hat{\beta}_i(t) = \beta_i - \beta_c$$

The CCS algorithm is repeated in synchronization rounds, which basically consists of two main tasks: offset compensation, and skew compensation.

**Offset compensation:** In this phase, nodes exchange local clock readings which are used to synchronize nodes to a common time. The goal is to remove the offset error from all the clocks in the network. For this, each node tries to accurately estimate the instantaneous average time of all the clocks in the network, as given below, and set their clocks to this time.

$$\overline{C}(t) = \frac{1}{N}\sum_{i=1}^{N} C_i(t)$$

**Skew compensation:** In this, the nodes iteratively compare the results from the current and previous synchronization round in order to improve their skew compensation parameter. The goal here is to ensure all compensated clocks in the network tick at the same rate. That means

$$\lim_{t\to\infty} \frac{\alpha_i}{\hat{\alpha}_i(t)} = \alpha_c, \quad \forall i$$

For perfect offset compensation the skew rate of the consensus clock is given by:

$$\alpha_c = \frac{1}{N}\sum_{i=1}^{N} \alpha_i$$

However in reality, packet losses & the random transmission order of the nodes affects the final consensus skew rate $\alpha_c$

**Table 1: A comparison chart of various schemes is shown above**

| Schemes | | Energy efficiency | Complexity | Scalability | Fault Tolerance |
|---|---|---|---|---|---|
| RBS | | high | high | good | no |
| TPSN | | high | low | poor | no |
| LTS | | high | low | average | no |
| FTSP | | high | high | average | no |
| Global clock synchronization in sensor networks | All-node-based | low | low | very poor | no |
| | Diffusion-based | average | high | good | yes |
| Fault tolerant FTSP | | average | high | average | yes |
| Consensus clock synchronization for WSN | | high | low | good | yes |

## CONCLUSION

Synchronization in a distributed system such as Wireless Sensor Network is necessary for its effective functioning. The problem of synchronization is not new. Several schemes have been proposed till now. This paper presents some of the synchronization schemes which have been proposed till date.

## REFERENCES

[1]. M. K. Maggs, S. G. Keefe, and D. V. Thiel, "Consensus Clock Synchronization for Wireless Sensor Networks," in Proc. IEEE Sensors Journal, vol. 12, no. 6, June 2012.
[2]. W. Su and I. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," IEEE/ACM Trans. Netw., vol. 13, no. 2, Apr 2005.
[3]. L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in Proc. 46th IEEE Conf. Decis. Control, New Orleans, LA, Dec. 2007.
[4]. Q. Li and D. Rus, "Global clock synchronization in sensor networks," in Proc. IEEE Conf. Comput. Commun., vol. 1. Hong Kong, China, Mar. 2004, pp. 564–574.
[5]. M. Maroti, G. Simon, B. Kusy, and A. Ledeczi, "The flooding time synchronization protocol," in Proc. 2nd Int. Conf. Embed. Netw. Sensor Syst., Baltimore, MD, Nov. 2004, pp. 39–49.
[6]. J. Elson, L. Girod, and D. Estrin, "Finegrained network time synchronization using reference broadcasts," in Proc. 5th Symp. Operat. Syst. Design Implement., vol. 36. Dec. 2002, pp. 147–163.
[7]. M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in Proc. IEEE Wireless Commun.Netw. Conf., Mar. 2003, pp. 1266–1273.
[8]. S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in Proc. 1st ACM Conf. Embed. Netw. Sensor Syst., Nov. 2003, pp. 138–149.
[9]. F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: Asurvey," IEEE Netw., vol. 18, no. 4, pp. 45–50, Jul.–Aug. 2004.
[10]. H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," IEEE Trans. Comput., vol. 36, no. 8, pp. 933–940 Aug. 1987.
[11]. J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in Proc. 15th Int. Parallel Distr. Process. Symp. Workshops, vol. 3. 2001, pp. 30186b-1–30186b-6.
[12]. S. Ganeriwal, R. Kumar, S. Adlakha, M. B. Srivastava, "Network-wide time synchronization in sensor networks," NESL Technical Report, 2003.
[13]. S. Ganeriwal, Mani B. Srivastava, "Timing-sync Protocol for Sensor Networks (TPSN) on Berkeley Motes", NESL 2003.
[14]. J. Elson, K. Romer, "Wireless Sensor Networks: A New Regime for Time Synchronization," Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey. October 28-29 2002.
[15]. D. L. Mills, "Internet time synchronization: The Network Time Protocol" In Z. Yang and T.A. Marsland, editors, Global States and Time in Distributed Systems. IEEE Computer Society Press, 1994.