TREE STRUCTURED TIME SYNCHRONIZATION PROTOCOL IN WIRELESS SENSOR NETWORK

Surendra Rahamatkar¹, Dr. Ajay Agarwal², Vineet Sharma³, Pankaj Gupta⁴

¹²³Krishna Institute of Engineering & Technology, Ghaziabad, India
⁴Ghyan Bharti Institute of Technology, Meerut, India
rahamatkar_s@rediffmail.com

ABSTRACT

Wireless sensor networks (WSNs) assume a collection of tiny sensing devices connected wirelessly and which are used to observe and monitor a variety of phenomena in the real physical world. Time synchronization is an important issue in wireless sensor networks. Many applications based on these WSNs assume local clocks need to be synchronized to a common view of clock at each sensor node. Some essential limitations of sensor networks such as limited energy resources, storage, computation, and bandwidth, combined with potentially high density of nodes make traditional synchronization methods incompatible for these networks. Hence, an increasing research focus on designing synchronization schemes is required. This paper reviews existing time synchronization protocols and the need for synchronization in sensor networks and then presents the proposed algorithm to construct adhoc tree structure of sensor network along with the process of clock synchronization.

Keywords- Wireless sensor networks, Clock synchronization, Tree Structure.

1. INTRODUCTION

As the advances in technology have enabled the development of tiny, low power devices capable of performing sensing and communication tasks, sensor networks emerged and received high attention of many researchers. Sensor networks are a special type of ad-hoc networks, where wireless devices (usually referred as nodes in the network) get together and spontaneously form a network without the need for any infrastructure.

Wireless sensor networks can be applied to a wide range of applications in domains as diverse as medical, industrial, military, environmental, scientific, and home networks [1]. Since the sensors in a wireless sensor network operate independently, their local clocks may not be synchronized with one another. This can cause difficulties when trying to integrate and interpret information sensed at different nodes. For instance, if a moving car is detected at two different times along a road, before we can even tell in what direction the car is going, the detection times have to be compared meaningfully. In addition, we must be able to transform the two time readings into a common frame of reference before estimating the speed of the vehicle. Estimating time differences across nodes accurately is also important in node localization. For example, many localization algorithms use ranging technologies to estimate internodes distances; in these technologies, synchronization is needed for time-of-flight measurements that are then transformed into distances by multiplying with the medium propagation speed for the type of signal used such as radio frequency or ultrasonic. There are additional examples where cooperative sensing requires the nodes involved to agree on a common time frame such as configuring a beam-forming array and setting a TDMA (Time Division Multiple Access) radio schedule [2]. These situations mandate the necessity of one common notion of time in wireless sensor networks. Therefore, currently there is a huge research interest towards developing efficient clock synchronization protocols to provide a common notion of time.

The clock synchronization problem has been studied thoroughly in the areas of Internet and local area networks (LANs) for the last several decades. Many existing synchronization algorithms rely on the clock information from GPS (Global Positioning System). However, GPS-based clock acquisition schemes exhibit some weaknesses: GPS is not ubiquitously available and requires a relatively high-power receiver, which is not possible in tiny and cheap sensor nodes. This is the motivation for developing software-based approaches to achieve innetwork time synchronization. Among many protocols that have been devised for maintaining synchronization in Computer Networks, NTP
(Network Time Protocol) [3] is outstanding owing to its ubiquitous deployment, scalability, robustness related to failures, and self-configuration in large multihop networks. Moreover, the combination of NTP and GPS has shown that it is able to achieve high accuracy on the order of a few microseconds [4]. However, NTP is not suitable for a wireless sensor environment, since wireless sensor networks pose numerous challenges of their own; to name a few, limited energy and bandwidth, limited hardware, latency, and unstable network conditions caused by mobility of sensors, dynamic topology, and multi-hopping. Hence, clock synchronization protocols different from the conventional protocols are needed in order to deal with the challenges specific to WSNs.

This paper consists of several sections. Section 2 contains theoretical clock model for sensor network. Section 3 contains the need for synchronization in Sensor Networks along with requirements on the synchronization schemes for Sensor Networks as its subsection and then provides the review of existing synchronization protocols in section 4. We present our proposed Tree Structured Time Synchronization scheme in section 5 then evaluation and comparison of the proposed scheme with related existing work is presented in section 6. Finally, section 7 contains the conclusion of the paper.

2. CLOCK MODEL

There have been various theoretical results that have been proven regarding clock synchronization. These analytical results and their consequences are useful when designing a clock synchronization protocol. From the causality property in a system, the ordering of events can be formally stated. If an event ‘a’ occurs before another event ‘b’, then ‘a’ should happen at an earlier time than ‘b’. Let Ci(a) be the clock value of process i when event ‘a’ occurs. Then it can be formally stated that:

If ‘a’ and ‘b’ are events in process i, and event ‘a’ occurs before event ‘b’, then

\[ Ci(a) < Ci(b) .\]

Lamport [5] showed that, when the value of a clock needs to be adjusted, it always has to be set forward and never back. Setting the clock back could cause the above condition to be violated. Hence, in an ideal system, the slower clocks needs to be adjusted to the value of the fastest clock, for all clocks to be synchronized. This restriction will also maintain the partial ordering of the events.

It is useful to have a bound on the best accuracy achievable in any system, such that no bound lower than that is specified. Srikanth et al. [6] have shown that for any synchronization algorithm, even in the absence of faults, the bound on the rate of drift of logical clocks from real time is greater than the bound on the rate of drift of physical clocks. In the presence of faults such as message losses and node failures, the accuracy of logical clocks becomes even worse.

2.1 A simple data collection algorithm:

Classical data collection algorithm [4] can be used to define the time synchronization, however in this paper we process the data stream differently. Consider two wireless nodes 1 and 2, with their hardware clocks \( t_1(t) \) and \( t_2(t) \) respectively, where \( t \) is the Universal Coordinated Time (UCT). In general the hardware clock of node \( i \) is a monotonically non-decreasing function of \( t \). In practice, a quartz oscillator is used to generate the real time clock. The oscillator’s frequency depends on the ambient conditions, but for relatively extended periods of time (minutes - hours) can be approximated with good accuracy by an oscillator with fixed frequency:

\[ t_i(t) = at_i + b_i; \]

where \( a \) and \( b \) are the drift and the offset of node \( i \)’s clock. In general \( ai \) and \( bi \) will be different for each node and approximately constant for an extended period of time.

From (1) it follows that \( t_i \) and \( t_j \) are linearly related:

\[ t_j(t) = a_{ij}t_i(t) + b_{ij} \]

The parameters \( a_{ij} \) and \( b_{ij} \) represent the relative drift and the relative offset between the two clocks respectively. If the two clocks are perfectly synchronized, the relative drift is equal to one and the relative offset is equal to zero.

Assume that node 1 would like to be able to determine the relationship between \( t_j \) and \( t_c \). Node 1 sends a probe message to node 2. The probe message is time stamped right before it is sent with \( t_a \). Upon receipt, node 2 timestamps the probe \( t_b \) and returns it immediately to node 1 which timestamps it upon receipt \( t_c \). Figure 1 shows such message exchange.

\[ Figure 1: A probe message from node 1 is immediately returned by node 2 and time stamped at each send/receive point resulting in the data-point \((t_a, t_b, t_c)\). \]

The three time-stamps \((t_a, t_b, t_c)\) form a data-point which effectively limits the possible values of parameters \( a \) and \( b \) in (2). Indeed, since \( t_a \) happened before \( t_b \) and \( t_b \) happened before \( t_c \) the following properties should hold:
3. NEED FOR TIME SYNCHRONIZATION

The clocks of each node in a WSN should read the same time within epsilon and remain that way. Since clocks drift over time, they must be periodically re-synchronized and in some instances when very high accuracy is required it is even important for nodes to account for clock drift between synchronization periods.

Clock synchronization is important for many reasons. When an event occurs in a WSN it is often necessary to know where and when it occurred. Clocks are also used for many system and application tasks. For example, sleep/wake-up schedules, some localization algorithms, and sensor fusion are some of the services that often depend on clocks being synchronized. Application tasks such as tracking and computing velocity are also dependent on synchronized clocks.

There are several reasons for addressing the synchronization problem in sensor networks. First, sensor nodes need to coordinate their operations and collaborate to achieve a complex sensing task. Data fusion is an example of such coordination in which data collected at different nodes are aggregated into a meaningful result. For example, in a vehicle tracking application, sensor nodes report the location and time that they sense the vehicle to a sink node which in turn combines this information to estimate the location and velocity of the vehicle. Clearly, if the sensor nodes lack a common timescale (i.e., they are not synchronized) the estimate will be inaccurate.

Second, synchronization can be used by power saving schemes to increase network lifetime. For example, sensors may sleep (go into power-saving mode by turning off their sensors and/or transceivers) at appropriate times, and wake up when necessary. When using power-saving modes, the nodes should sleep and wake-up at coordinated times, such that the radio receiver of a node is not turned off when there is some data directed to it. This requires a precise timing between sensor nodes.

Scheduling algorithms such as TDMA can be used to share the transmission medium in the time domain to eliminate transmission collisions and conserve energy. Thus, synchronization is an essential part of transmission scheduling.

Traditional synchronization schemes such as NTP or GPS are not suitable for use in sensor networks because of complexity and energy issues, cost and size factors. NTP works well synchronizing the computers on the Internet, but is not designed with the energy and computation limitations of sensor nodes in mind. A GPS device may be too expensive to attach on cheap sensor devices, and GPS service may not be available everywhere, such as inside the buildings or under the water.

Furthermore in adversarial environments, the GPS signals may not be trusted.

3.1 Requirements on the Synchronization Schemes for Sensor Networks

In this section we present a broad set of requirements for the synchronization problem. These requirements can also be regarded as the metrics for evaluating synchronization schemes on sensor networks. However, there are trade offs between the requirements of an efficient synchronization solution (e.g., precision versus energy efficiency), thus a single scheme may not satisfy them altogether.

Energy Constraint: Energy efficiency is very important for sensor networks as opposed to traditional networks.

Scalability: Most sensor network applications need deployment of a large number of sensor nodes. A synchronization scheme should scale well with increasing number of nodes and/or high density in the network [9].

Tunable Accuracy: Traditional time synchronization protocols try to achieve the highest degree of accuracy possible. The higher the level of accuracy required, the higher the resource requirement. The accuracy of required synchronization depends on the application requirement. Therefore, there is a need for a trade-off between resource requirements and accuracy, depending on the need of the application and resource availability of the system.

Non-determinism: Sensor networks are dynamic systems with considerably higher rate of failures and non-determinism of the individual nodes than in tradition networks. Thus the synchronization protocol needs to be more robust to failures and also to the greater variability in communication delay.

Cost and Size: Wireless sensor nodes are very small and inexpensive devices. Therefore, as noted earlier, attaching a relatively large or expensive hardware (such as a GPS receiver) on a small, cheap device is not a logical option for synchronizing sensor nodes. The synchronization method for sensor networks should be developed with limited cost and size issues in mind.

Server-less: Traditional protocols have specified servers, with multiple accuracy levels which are sources of accurate time. Sensor networks do not have any external infrastructure present and can be large in scale. Maintaining global time scale in this network is thus harder, if no external broadcast source of global time such as GPS is present. Elson et al. [10] proposed that each node maintain an undisciplined clock, augmented with the relative frequency and phase information of its neighbors.
4. EXISTING APPROACHES TO TIME SYNCHRONIZATION

Time synchronization algorithms providing a mechanism to synchronize the local clocks of the nodes in the network have been extensively studied in the past. The most widely adapted protocol used in the internet domain is the Network Time Protocol (NTP) devised by Mills [13]. The NTP clients synchronize their clocks to the NTP timeservers with accuracy in the order of milliseconds by statistical analysis of the round-trip time. The timeservers 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.

Two of the most prominent examples of existing time synchronization protocols developed for the wireless sensor network domain are the Reference Broadcast Synchronization (RBS) algorithm [10] and the Timing-sync Protocol for Sensor Networks (TPSN) [11].

**RBS:** In the RBS, a reference message is broadcasted. The receivers record their local time when receiving the reference broadcast and exchange the recorded times with each other. The main advantage of RBS is that it eliminates transmitter-side non-determinism. The disadvantage of the approach is that additional message exchange is necessary to communicate the local time-stamps between the nodes. To our best knowledge the algorithm has not been extended to large multi-hop networks.

**TPSN:** The TPSN algorithm first creates a spanning tree of the network and then performs pair wise synchronization along the edges. Each node gets synchronized by exchanging two synchronization messages with its reference node one level higher in the hierarchy. The TPSN achieves two times better performance than RBS by time-stamping the radio messages in the Medium Access Control (MAC) layer of the radio stack [11] and by relying on a two-way message exchange. The shortcoming of TPSN is that it does not estimate the clock drift of nodes, which limits its accuracy, and does not handle dynamic topology changes.

5. TREE STRUCTURED TIME SYNCHRONIZATION PROTOCOL (TSTP)

In this section we present a Tree Structured Time Synchronization protocol, which works on two phases. First phase used to construct a adhoc tree structure and second phase used to synchronize the local clocks of sensor nodes.

The goal of the TSTP is to achieve a network wide synchronization of the local clocks of the participating nodes. We assume that each node has a local clock exhibiting the typical timing errors of crystals and can communicate over an unreliable but error corrected wireless link to its neighbors. The TSTP synchronizes the time of a sender to possibly multiple receivers utilizing a single radio message time-stamped at both the sender and the receiver sides. MAC layer time stamping can eliminate many of the errors, as observed in [12] and [11]. However, accurate time-synchronization at discrete points in time is a partial solution only. Compensation for the clock drift of the nodes is inevitable to achieve high precision in-between synchronization points and to keep the communication overhead low. Linear regression is used in TSTP to compensate for clock drift as suggested in [10].

5.1 Adhoc Tree Structure Construction phase

Before the sensors can be synchronized, a network topology based on tree structure must be created.

**Algorithm 1:** Tree structure construction

```
Begin
Accept (fd_pckts)
Initialize : no_reciever = 0;
If (current_reciever = = root)
  Broadcast (fd_pckts)
Else if (current_reciever != root)
Begin
Accept (fd_pckts);
Source(broadcast_msg) = Parent
(curent_reciever);
Node_Level (current_reciever) =
  Node_Level (Parent)+1;
Broadcast (ack_pckt, node_id);
Ignore (fd_pckts);
End
Else if (current_node receives ack_pckt)
no_receiver++;
End
```

Algorithm 1 is used by each sensor node to efficiently flood the network to form a hierarchical structure from a designated source point. Each sensor is initially set to accept fd_pckts (flood packets) for first time, but will ignore subsequent ones in order not to be continuously reassigned as the flood broadcast propagates.

When a node receives or accepts the fd_pckts then first it set to its parent as source of broadcast after that level of current receiver node will be assigned one more than the level of parent node and then it broadcast the fd_pckts along with node identifier and level. The no_reciever variable keeps track of the node’s receivers.
5.2 Time Synchronization phase

In this section we are presenting a Tree Structured Synchronization Protocol, which is based on the protocol proposed by [11], that the aim is 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.

![Figure 2: Two way message exchange between a pair of nodes.](image)

This proposed protocol used for multihop synchronization of the network based on pair wise synchronization scheme of [11]. This requires nodes to synchronize to some reference point(s) such as a sink node in the sensor network and needs a tree to be constructed first. Then pair wise synchronization is done along the n - 1 edges of the tree. In this centralized algorithm, the reference node is the root of the tree and has the responsibility of initiating a “resynchronization” when required. 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 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.

The basic building block of the synchronization process is the two-way message exchange between a pair of nodes. Here we assume that the clock drift between a pair of nodes is constant in the small time period during a single message exchange. The propagation delay is also assumed to be constant in both directions. Consider a two-way message exchange between nodes A and B as shown in figure 2. Node A initiates the synchronization by sending a synchronization-msg at time \( t_j \) as per node’s local clock. This Message includes A’s identity, and the value of \( t_j \). B receives this message at \( t_j \) which can be calculated as:

\[
t_j = t_j + \Delta + d,
\]

where \( \Delta \) is the relative clock drift between the nodes, and \( d \) is the propagation delay of the pulse. B responds at time \( t_j \) with an acknowledgement, which includes the identity of B and the values \( t_j, t_j \), and \( t_j \). Then, node A can calculate the clock drift and propagation delay as below, and synchronize itself with respect to node B.

\[
\Delta = \frac{(t_j - t_j) - (t_j - t_j)}{2}
\]

\[
d = \frac{(t_j - t_j) + (t_j - t_j)}{2}
\]

The synchronization phase is initiated by the root node’s timesync message. On receiving this message, nodes of level 1 initiate a two-way message exchange with the root node. Before initiating the message exchange, each node waits for some random time, in order to minimize collisions on the wireless channel. Once they get back a reply from the root node, they adjust their clocks to the root node. Level 2 nodes, overhearing some level 1 node's communication with the root, initiate a two-way message exchange with a level 1 node, again after waiting for some random time to ensure that level 1 nodes have completed their synchronization. This procedure eventually gets all nodes synchronized to the root node.

6. EVALUATION AND COMPARISON

In this section we are presenting the advantages and disadvantages of the proposed protocol (TSTP) compared to the well known protocol like Reference Broad Cost (RBS). The ideas from TPSN protocol were used to enhance our proposed protocol. This time synchronization protocol was also developed with the special requirements of sensor networks in mind as opposed to other, more general algorithms.

The proposed pair wise synchronization scheme based on [11] eliminates the access time, byte alignment time and propagation time by making use of the implicit acknowledgments to transmit information back to the sender. This protocol gains an additional accuracy over RBS due to time-stamping the radio message multiple times and averaging these time-stamps.

The classical sender-receiver pair wise synchronization scheme is implemented on Berkeley’s Mica architecture [14], and makes use of time stamping packets at the MAC layer in order to reduce uncertainty at sender. Ganeriwal et.al. [11] claim that pair wise synchronization scheme achieves two times better precision than RBS. They state that the precision of 6.5µs reported for RBS is due to using a superior operating system (Linux).

Thus RBS is implemented on Mica sensor architecture, as well as pair wise synchronization scheme in order to compare their performance. RBS has actually been tested on Berkeley motes (by Elson et.al.) [10], and the reported precision was 11µs. However, Ganeriwal et.al. report [11], on average, 29.13µs precision for their implementation of RBS on Mica. The average error of pair wise synchronization scheme is 16.9µs with its implementation on the same hardware platform.
Essentially it is claimed that uncertainty at the sender contributes very little to the total synchronization error, as it is minimized by the use of low level timestamps at the sender, and therefore the classical sender-receiver synchronization is more effective than receiver-receiver synchronization in sensor networks.

7. CONCLUSION

Wireless sensor networks have tremendous advantages for monitoring object movement and environmental properties but require some degree of synchronization to achieve the best results. In this paper we have proposed a Tree Structure based Time Synchronization Protocol. The proposed protocol is able to produce tight, deterministic synchronization with only few message exchanges.

While the proposed protocol is suitable for any type of network, it is especially useful in wireless sensor networks which are typically extremely constrained on the available computational power and bandwidth and have some of the most exotic needs for high precision synchronization. The proposed synchronization protocol was designed to switch between Timing-sync Protocol for Sensor Networks (TPSN) and the Reference Broadcast Synchronization algorithm (RBS). These two algorithms allow all the sensors in a network to synchronize themselves within a few microseconds of each other, while at the same time using the least amount of resources possible. In proposed work two varieties of the algorithm are presented in and their performance is verified theoretically with the existing results and compared with existing protocols.

8. REFERENCES