Time Synchronization in Wireless Sensor Networks: A Survey

Sami M. Lasassmeh and James M. Conrad
Electrical and Computer Engineering Department, University of North Carolina at Charlotte, NC, USA
{slasassm, jmconrad}@uncc.edu

Abstract — **Wireless sensor networks consist of small devices distributed over geographical area. Each one of these devices has sensing, computing, and communicating components. Wireless sensor networks are used in many applications where partial or full time synchronization in the network is required. Time synchronization aims at equalizing the local times for all nodes in the network, if necessary. Since wireless sensor networks are limited in energy resources, computation capability, storage capacity, and bandwidth, traditional time synchronization algorithms like Network Time Protocol (NTP) and Global Positioning System (GPS) are impractical to synchronize the network. This paper explains the time synchronization problem in wireless sensor networks and details the basic algorithms proposed in this area.**

I. INTRODUCTION

A wireless network consists of spatially distributed autonomous devices to cooperatively monitor physical or environmental conditions such as temperature, sound, vibration, pressure, motion and pollutants, at different locations [2, 10]. Each device is equipped with sensing components, a transceiver for wireless communication, computing circuitry for processing and networking of the data, and an energy source, usually a battery [4].

Sensor networks are a special type of ad-hoc network, where wireless devices (usually referred to as nodes in the network) work together to spontaneously form a network without the need for any infrastructure [12]. Nodes can send data, receive data, or act as routers in the network.

Network nodes are usually static and resource constrained. They collect the data and forward it to a central unit, called a sink, for further processing. The life time of the whole wireless sensor network, which is defined as the time for percentage of wireless nodes to run out of power, not the life time of one device, is one of the main parameters when evaluating network performance.

Sensors must be designed to withstand harsh environmental and geographical conditions. Usually extra devices or sensors are deployed to achieve redundancy and to compensate for devices which can fail (i.e. hardware failure, total consumption of battery power).

Time synchronization, which aims at creating a common time scale in the network, is a necessity for many applications. Medium access control techniques like TDMA require time synchronization for an accurate scheduling of medium accesses with no collisions. In addition, wireless sensor networks have limited energy resources; therefore power-saving techniques are used to lower the power consumption of whole network. In order for the nodes to turn their transceivers on and off at appropriate times, precise timing between network nodes is required. In many applications, data fusion is used. Data collected at different nodes are aggregated and sent to a sink node to formulate a decision. These nodes collaborate and coordinate with one another to perform the task. Those kinds of applications, like target detection; require precise timing between nodes in order to get an accurate estimation of the event.

Traditional synchronization techniques that have been used in wired networks are not suitable for wireless networks. NTP [8], which is widely used in the Internet, is too complicated to implement and is not energy efficient. GPS [5], on the other hand, is not practical since the device that is needed to be attached to the network node is large and costly. Also, since sensors are usually deployed in harsh environments, GPS signals are often inaccessible.

The following sections are organized as follows: In Section II, we describe how computer clocks work and explain sources and requirements of time synchronization. Section III details main time synchronization methods; Traditional Time Synchronization (TTS), Reference Broadcast Synchronization (RBS), Timing-Sync Protocol for Sensor Networks (TPSN), Tiny-Sync and Mini-Sync, and Tree-based Synchronization Algorithms. Section IV concludes the paper by comparing different synchronization methods.

II. SYNCHRONIZATION ISSUES

A. Computer clocks

Computer clock circuits consist of an oscillator and a counter. Based on the oscillator angular frequency, the counter increases its value to represent the local clock C(t) of a network node. In ideal situations, angular frequency is

978-1-4244-5853-0/10/$26.00 ©2010 IEEE

constant. Thus, the clock rate of change $\frac{dC}{dt}$ is equal to 1. Due to physical variations, like temperature, vibration, and pressure, the angular frequency changes and computer clocks drifts. The local clock of node *i* can be related to real time *t* as follows [11]:

$$C_i(t) = a_i t + b_i \quad (1)$$

Where a_i is the clock *drift,* and b_i is the *offset* of node *i*'s clock. *Drift* denotes the rate (frequency) of the clock, and *offset* is the difference in value from real time *t*. Using equation (1), we can compare the local clocks of two nodes in a network, say node 1 and node 2 as in [11]:

$$C_1(t) = a_{12} . C_2(t) + b_{12} \quad (2)$$

We call a_{12} the *relative drift* and b_{12} the *relative offset* between the clocks of node 1 and node 2. If two clocks are perfectly synchronized, then their relative drift is 1, meaning the clocks have the same rate. Their relative offset is zero, meaning they have the same value at that instant [12].

Based on the previous equations, clock rate of network node and offset can be used to adjust its local time. Some schemes designed to adjust offsets of nodes repeatedly, or adjust offset and clock rate to a common time scale.

B. Sources of time synchronization errors

Message exchange is used in many time synchronization algorithms. If one node sends a packet with a time stamp, non deterministic delays like access and propagation times make it hard for the receiver node to synchronize precisely with the sender node.

In general, the following elements contribute to the synchronization errors [9]:

- *Send time:* This is the total time of building the message and transfer it to the network interface to be sent. This time highly depends on the operating systems in use.
- *Access time:* This is the time needed to access the channel. Every network employs a medium access control (MAC) scheme, like time division multiple access (TDMA), and total access time depends on that scheme. In TDMA for example, network node has to wait for its slot to start transmitting while in other schemes, network nodes wait for the channel to be idle.
- *Propagation time:* This is the time required to propagate the message through the air from network interface of the sender to the network interface of the receiver.
- *Receive time:* This is the time spent in receiving the message by network interface and transferring it to the application layer of the host.

C. Requirements on the Synchronization schemes for Sensor Networks

When designing time synchronization algorithm, wireless network limitations enforce certain requirements that need to be met. Usually, the following metrics are used to evaluate any synchronization technique:

- *Accuracy:* Precision of synchronization technique highly depends on the application.
- *Robustness:* Network nodes might die or go out of scope because of the harsh environment they are deployed in. Any synchronization scheme should adapt to such changes in the network and function in all situations.
- *Scalability:* In some applications, tens of thousands of sensors might be deployed. Any synchronization technique must work well with any number of nodes in the network.
- *Longevity:* Based on the application, provided time synchronization may be instantaneous, e.g. when a certain event happens, or may last as long as the network operates.
- *Energy efficiency:* Network nodes have limited energy resources. All network protocols including synchronization ones should consider this limitation.
- *Cost:* Due to advanced technologies, network nodes are becoming so small and inexpensive. Any synchronization algorithm should not add cost or increase the size of network node.
- *Scope:* Time synchronization algorithm provides a common time for all nodes in the network, which costs more energy and time, or provides a common time to only spatially close nodes.
- *Delay:* Many applications, like detecting gas leak, require an immediate response. For those kinds of applications, the total required time to synchronize the network must be as low as possible.

III. SYNCHRONIZATION METHODS FOR WIRELESS SENSOR NETWORKS

Synchronization schemes aim at adjusting the local times of network nodes to the same reference value. The most strict and power consuming schemes require synchronization of all nodes in the network at all times (always-on), while other more relaxed schemes require synchronization of few nodes at a time.

A. Traditional Time Synchronization (TTS)

Traditional schemes to synchronize network nodes, like Network Time Protocol (NTP), are based on sending and receiving messages. In the simplest form, sender transmits a message with its current local time to the receiver. Then the receiver adjusts its clock to the received time. This scheme works if the delay between sending and receiving messages is negligible compared to the total desired accuracy. If the delay is large, node sends a message and assuming that the receiver replies back instantaneously; node can calculate the round trip time and use that time to synchronize with the other nodes.

B. Reference Broadcast Synchronization (RBS)

In [1], RBS was introduced, where a receiver with receiver, instead of sender with receiver, synchronization is used. Nodes broadcast beacons and other nodes use the arrival time of those beacons as a reference to find the time offsets between them. The main idea in this algorithm is to remove the nondeterministic of the send time. The only error sources in this scheme are due to the nondeterministic of propagation and receive times. Errors due to propagation time can be neglected assuming that the radio range is short

and that the beacon is broadcasted to all nodes at this time. So receive time is the only source of error in this scheme.

Typical communication scenario is that one node sends a beacon to its neighbors, and receivers exchange their receive time of this beacon to find their relative time offsets, and hence synchronize with each others. Precision of this scheme increases with the increase number of beacons used to synchronize.

Precision of few μs can be reached using this technique. On Berkeley mote, which is a popular platform, a precision of 11 μs was achieved. Also, this algorithm was applied to Linux machines communicating through the 802.11 protocol and a precision of 6.29±6.45 μs was reached.

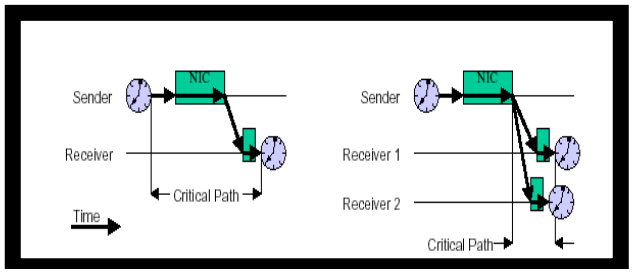

Figure 2. A critical path analysis for traditional time synchronization protocols (*left*) and RBS (*right*) [1].

C. Timing-Sync Protocol for Sensor Networks (TPSN)

TPSN is proposed in [6]. This synchronization technique consists of two phases; level discovery phase and synchronization phase. In the discovery phase, each node assigned a level; only one node is assigned level 0 and it is called the root node. In the second phase, a node of level *n* synchronizes to a node of level *n-1* and by the end of this phase, a network-wide synchronization is achieved.

- *Level discovery phase:* The first step in this level is selecting the root node. Nodes in the network can periodically elect this node, and level 0 is assigned to it. This node can be connected to an external time reference, like GPS. Then this node sends level discovery packets and all neighbor nodes which receive this packet assign themselves level 1. Nodes continue to send packets which include their level number and identities until all nodes in the network are assigned a level.
- *Synchronization phase:* Again, the root node starts the synchronization phase by sending time_sync packet. Node A of level *n* synchronizes to node B of level *n-1* through a two-way message exchange. Node A sends a packet with it is local send time T_1. Node B receives the packet at time T_2, which can be calculated as [6]:

$$T_2 = T_1 + T_d + \Delta \quad (3)$$

Where T_d is the propagation delay and Δ is the relative clock drift between the nodes and both assumed to be constant within the time of exchanging messages. Node B waits for a random time and responds back to node A through an acknowledgment packet at time T_3, which includes the values of T_1, T_2, T_3, and its level number. Once node A receives this packet at T_4, it can calculate Δ and T_d as follows [6] and synchronize itself to node B.

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2}; \quad T_d = \frac{(T2 - T1) + (T4 - T3)}{2} \quad (4)$$

This method of synchronizing all nodes in level *n* to nodes in level *n-1* continues until all nodes in the network get synchronized.

Comparing TPSN to RBS, both algorithms were applied on Mica platform. It was reported that RBS achieved a precision of 29.13 μs, while TPSN achieves16.9 μs for the same experimental setup. Those results show that traditional sender-receiver algorithm can overcome send time errors using time stamps at MAC layer and can perform better in terms of precision than receiver-receiver algorithms.

In terms of energy, TPSN has more overhead than RBS. Packets sent to select a root node and those sent in the level discovery phase are considered an overhead. This sort of overhead is shortening network lifetime, especially if the root nodes die so often in the network and the whole synchronization method must be repeated.

D. Tiny-Sync and Mini-Sync

In [11], authors propose two simple algorithms to synchronize a pair of wireless nodes. Then those algorithms are extended to synchronize the whole network nodes. Traditional two-way messaging is used to collect data points, which are used to apply tight bounds on relative drift and relative offset between two nodes. To create data point using two nodes; 1 and 2, node 1 sends a probe message with timestamp t_o. Node 2 timestamps the received message with t_b and sends back an acknowledgment to node 1, immediately or after sometime, which timestamps this acknowledgment with t_r. Fig. 3 shows the previous described sequence.

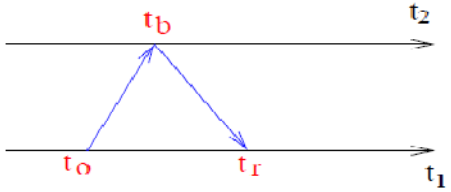

Figure 3. Probe message exchange [11].

Based on equation 2 above and taking into account the order of transferred messages, the following relationships should hold [11]:

$$t_o(t) < a_{12} t_b(t) + b_{12} \quad (5)$$

$$t_r(t) > a_{12} t_b(t) + b_{12} \quad (6)$$

Algorithms start collecting data points, using linear programming to estimate a_{12} and b_{12}, and synchronizing nodes with each others. While regular data and acknowledgment packets in the network can be used to collect data points, which reduce the overhead, estimating a_{12} and b_{12} require large storage and heavy computation.

E. Tree-based Synchronization Algorithms

Some wireless sensor networks need a precision in order of micro seconds, while others need it in order of milliseconds. The algorithms proposed in [7], deal with this fact by assuming that the precision to be achieved is given,

and hence, the complexity and computational time depend on this given value.

First algorithm consists of two phases; a spanning tree is constructed in phase one. Then a sender to receiver synchronization is done along the *n-1* edges of the spanning tree in phase two. In this centralized algorithm, all nodes synchronize to a central one, called sink or root node. Root node is in charge of initiating the synchronization phase when it is required. Also, it sets the total time to synchronize the whole network. It is a necessity to periodically update the root node with the spanning tree depth in order for it to make those calculations and other decisions.

In the second algorithm, nodes decide when they need to be synchronized based on their distance from the reference node, clock drift, and the required precision. So, any node in the network can initiate the synchronization phase with a reference point, and all nodes along the path to that reference point must be synchronized first. As some nodes do not need to be synchronized all the times, this algorithm removes the overhead of unnecessary synchronization by allowing nodes to make decisions. Aggregation of synchronization requests is used to make this algorithm more efficient. Before the node sends synchronization requests, it checks with its neighbors to see if any request is pending, if any, it will add its request to the pending one, so the path will be synchronized only once.

In [11], suggested algorithm divides the whole network into sub trees. In each sub tree, children synchronize to their parent or root node. Then, all parents synchronize with each others.

IV. CONCLUSION

We found that RBS and TPSN perform very well in terms of accuracy. Their accuracy is in order of few micro seconds. Since sending and receiving packets is the most power consuming task while synchronizing a network, number of packets required to synchronize a pair of nodes can be used to estimate power efficiency.

In RBS, which is a receiver to receiver algorithm, 4 packets sent and 3 packets received to synchronize two nodes. In TPSN, 2 packets sent and 2 packets received to synchronize two nodes. Although TPSN sounds more power efficient, the fact that extra sent and received packets are needed for level discovery in TPSN, and the fact that more than one receiver can be synchronized in one pulse in RBS.

Tree-based Synchronization Algorithms are flexible, and based on the given precision, complexity might be high or low. Root node plays a main role in those algorithms, and could be connected to external nodes to get the real life time.

REFERENCES

[1] J. Elson, L. Girod, and D. Estrin, “Fine-Grained Time Synchronization using Reference Broadcasts," Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.

[2] T. Haenselmann. “Sensornetworks,” Wireless Sensor Network textbook, 2006.

[3] L. He and G.S. Kuo, “A Novel Time Synchronization Scheme in Wireless Sensor Networks,” Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd Volume 2, 7-10 May 2006 pps. 568 – 572.

[4] J. Hill and D. Culler, “A Wireless Embedded Sensor Architecture for System-Level Optimization," Technical Report, U.C. Berkeley, 2001.

[5] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. “Global Positioning System: Theory and Practice,” Text Book, Fifth edition, 2001.

[6] S. Ganeriwal, R. Kumar, and M. Srivastava, “Timing Sync Protocol for Sensor Networks," ACM SenSys, Los Angeles, November 2003.

[7] 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.

[8] D. Mills, “Network Time Protocol (Version 3) Specification, Implementation and Analysis,” Technical Report, University of Delaware, 1992.

[9] S. Ping, “Delay Measurement Time Synchronization For Wireless Sensor Networks,” Technical Report, Intel Research Berkeley Lab, 2003.

[10] K. Römer and F. Mattern. "The Design Space of Wireless Sensor Networks," Wireless Communications, IEEE Volume 11, Issue 6, Dec. 2004 pps.54 – 61.

[11] M.L. Sichitiu and C. Veerarittiphan, “Simple, Accurate Time Synchronization for Wireless Sensor Networks,"Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE Volume 2, 20-20 March 2003 pps.1266 - 1273 vol.2.

[12] F. Sivrikaya and B. Yener, “Time Synchronization in Sensor Networks: A Survey,” Network, IEEE Volume 18, Issue 4, July-Aug. 2004, pps.45 – 50.