DATS: A Distributed Algorithm for Time Synchronization in Wireless Sensor Network

Farzad kiyani[1], Ali Aghaee_rad[2], Hamidreza Tahmasebi_rad[2]

[1]Computer engineering Department of Islamic Azad university of Shabestar
[2]Computer engineering Department of Islamic Azad university of Zanjan

Abstract
In this paper, for decrease transmissions delay in wireless sensor network, a distributed algorithm (DATS) suggested. Because this network have got limited energy, therefore the energy issue is very important. The exist protocols due to have transmission delay in sending data among nodes cause increase consuming energy in network. DATS because of is distributing solve the problem. In this algorithm, a distinguished node periodically sends time values along a spanning tree structure. One of the most important advantages of the DATS is purely distributed, local, and does not depend on a fixed structure for disseminating time values. In conclusion, results gained shows that suggested protocol decrease transmissions delay of wireless network above 30 percent compared to the exist approaches.

1. Introduction

The Sensor networks have provided us with several applications in wireless environments but have more problems that must solve their. Time synchronization is a critical piece of infrastructure in any distributed system. In sensor networks, a confluence of factors makes flexible and robust time synchronization particularly important, while simultaneously making it more difficult to achieve than in traditional networks. Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make particularly extensive use of synchronized time: for example, to integrate a time-series of proximity detections into a velocity estimate [2]; to measure the time-of-flight of sound for localizing its source [4]; to distribute a beam forming array [7]; or to suppress redundant messages by recognizing that they describe duplicate detections of the same event by different sensors [5]. Sensor networks also have many of the same requirements as traditional distributed systems: accurate timestamps are often needed in cryptographic schemes, to coordinate events scheduled in the future, for ordering logged events during system debugging, and so forth.
Wireless sensor network (WSN) are large-scale distributed systems, yet their unique characteristics, especially the severe resource constraints, require the reevaluation of traditional distributed algorithms for problems once considered to be solved. One of the basic middleware services of sensor networks is time synchronization. Time synchronization is required for consistent distributed sensing and control. Furthermore, common services in WSN, such as coordination, communication, security or distributed logging also depend on the existence of global time.
The many uses of synchronized time in a sensor network make it critical. However, the diversity of these roles also makes synchronization a difficult problem to solve. Application requirements vary widely on many axes, such as precision, lifetime, scope, availability, and energy budget. For example, acoustic applications require precision of several microseconds, while sensor tasking works on the time scale of hours or days. Local collaborations often require only a pair of neighbors to be synchronized, while global queries require global time. Event triggers may only require momentary synchronization, while data logging or debugging often require an eternally persistent timescale. Communication with a user requires a external, human timescale, whereas only relative time is important for purely in-network comparisons. Some nodes have large batteries and run all the time; others are so constrained that they only wake up occasionally, take a single sensor reading, and transmit it before immediately returning to sleep. A paradox of sensor networks, then, is that they make stronger demands on a time synchronization system than traditional distributed systems, while simultaneously limiting the resources available to achieve it. This paradox has made current synchronization schemes inadequate to the task. For solve problems, distributed algorithm based on tree structure for synchronization in wireless sensor network suggested.
The rest of this paper is organized as follows. Section 2 reviews the existing protocols for time synchronization. Section 3, introduces the proposed algorithm in detail. Simulation results and finally conclusions and future works are presented in End Sections.

Manuscript received February 5, 2011
Manuscript revised February 20, 2011

2. Related Work

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 [6]. 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.

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 [1] and the Timing-sync Protocol for Sensor Networks (TPSN) [3]. 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. 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 [3] 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.

A number of other groups have considered synchronized time in sensor networks, concurrently with our work. In [8] suggests that, for some applications, the time of events can be described in terms of their age rather than as an absolute time [8]. In essence, when two nodes exchange a message that describes an event in terms of is age, the time at which the message itself is sent becomes a common frame of reference for time. The notion of "now" at the instant a message is sent is inexact, due to nondeterministic and asymmetric latencies; this is an important source of error. In the long term, error is dominated by frequency differences among nodes; the effect is magnified as timestamps age. In [8] analysis of his scheme concludes it has a nominal precision of 1ms. In [10] describe the Time Diffusion Protocol (TDP), for achieving global time synchronization in sensor networks [10]. Their work has a number of strengths, including an automatic self-configuration through dynamic topology discovery. In addition, they quantitatively analyze the energy cost of their scheme. However, to date, TDP has not been implemented, and the authors' simulations leave out certain details-such as a realistic model of the channel's non-determinism-that are critical to understanding its expected performance.

So far, we have described schemes that produce various forms of clock agreement among a distributed set of nodes. Implicit to these schemes is the existence of a clock local to each node in the system. A clock, in essence, is a device that counts events indicating the passage of some time interval (e.g., 1 second), and uniquely labels each of these intervals. The events are generated by a frequency standard, based on the observation of a physical process, such as the transit of the Sun through a local meridian, the vibration of a quartz crystal, the beat of a pendulum, or the resonance of a cesium beam tube. The quality of a clock usually boils down to its frequency stability-that is, the ability of its frequency standard to emit events at a constant frequency over time. The absolute value of the frequency compared to the desired value-or, its frequency accuracy-is also important, but calibration can easily compensate for an inaccurate but stable standard.

The focus of our work is on methods of clock synchronization, not on construction of better clocks. However, clocks are very important: the error bound achieved by a clock synchronization method is linked to both the error inherent in the method itself, and the stability of the clocks' frequency standards. In fact, to some extent, the two are interchangeable. Stable clocks can compensate for a synchronization channel between them that is prone to large (but unbiased) errors: many synchronization events can be averaged over a long time. Similarly, a precise synchronization channel can compensate for a poor-stability frequency standard; frequent synchronization minimizes the time in-between when the clock is left to fend for itself. Many types of frequency standards exist. In general, as their stability and accuracy increase, so do their power requirements, size, and cost, all of which are important in sensor networks. Most commonly found in computer clocks are quartz crystal oscillators, characterized in [11]. Quartz crystals are attractive because they are inexpensive, small, use little power, and perform surprisingly well despite their low resource requirements.

In this paper, an algorithm by high accuracy in calculating time proposed that in section 3 presented.

3.Introduce Proposed Protocol

As described in the derivation for the influence field and the size of the classifier window, we require that observations occurring at the same physical time need to be time stamped with values differing by not more than 5 ms in order to achieve the desired classification accuracy. This imposes an accuracy requirement on the time synchronization service, i.e., the time values of any two nodes in the network cannot differ by more than 5 ms at any time instant, which translates to a per-hop accuracy of less than 500 ls. An important requirement of the time synchronization service is that it should be robust to node and link failures. Our first implementation of time synchronization was based on the traditional spanning tree based algorithm for synchronizing clocks. In this algorithm, a distinguished node periodically sends time values along a spanning tree structure. However, this algorithm, while providing good accuracy, suffers from the drawback of depending on a relatively static structure for time dissemination and hence performs poorly in the presence of unreliable nodes and links and network partitions. For this reason, we designed a truly distributed time synchronization service that is robust to these commonly occurring faults in sensor networks.

4. A Distributed Algorithm for Time Synchronization in Wireless Sensor Network

The basic idea behind our time synchronization algorithm is that of locally synchronizing to the fastest clock. In this scheme, each node maintains a local clock which is never changed and an estimate of network time which is stored as an offset with respect to the local clock. Each node broadcasts its network time value periodically and receives time values from its neighbors. If a node receives a time value greater than its own, it adjusts its local time to that value; otherwise it ignores the value received. Thus, the entire network synchronizes to the maximum clock value in the network. This scheme also guarantees that the timestamp values at every node are monotonically increasing.

This time synchronization algorithm is purely distributed, local, and does not depend on a fixed structure for disseminating time values. Consequently, even if a single node or link fails, the other nodes can receive time values from the rest of their neighbors. The protocol is thus robust to individual node and link failures. Node joins are also easy to handle as the new nodes have lower time values and hence catch up with the rest of the network and do not force other nodes to roll back their clocks. The algorithm can also handle network partitions in the sense that the nodes in each partition synchronize to the maximum time value in that partition. These properties of robustness to network dynamics like failures, joins, and partitions make this protocol suitable for mobile environments as well. We demonstrated the robustness of the time synchronization scheme in a mobile setting by partitioning a group of sensor nodes into two subgroups, each of which synchronized to the maximum value in its partition. We showed that whenever a node moved from one partition to another, it would either catch up with the time in the new partition or the new partition would catch up with the moving nodes time and consequently the original partition. In the presence of node mobility in both directions, the protocol converges to synchronize both partitions with high accuracy even if no two nodes in either partition can communicate with each other directly.

The DATS utilizes a radio broadcast to synchronize the possibly multiple receivers to the time provided by the sender of the radio message. The broadcasted message contains the sender's time stamp which is the estimated global time at the transmission of a given byte. The receivers obtain the corresponding local time from their respective local clocks at message reception. Consequently, one broadcast message provides a synchronization point (a global-local time pair) to each of the receivers. The difference between the global and local time of a synchronization point estimates the clock offset of the receiver. As opposed to the RBS protocol [1] and FTSP [12], the time stamp of the sender must be embedded in the currently transmitted message. Therefore, the time-stamping on the sender side must be performed before the bytes containing the time stamp are transmitted. Message broadcast starts with the transmission of preamble bytes, followed by SYNCH bytes, then with a message descriptor followed by the actual message data, and ends with CRC bytes. During the transmission of the preamble bytes the receiver radio synchronizes itself to the carrier frequency of the incoming signal. From the SYNCH bytes the receiver can calculate the bit offset it needs to reassemble the message with the correct byte alignment. The message descriptor contains the target, the length of the data and other fields, such as the identifier of the application layer that needs to be notified on the receiver side. The CRC bytes are used to verify that the message was not corrupted. The message layout is summarized in Figure 1.

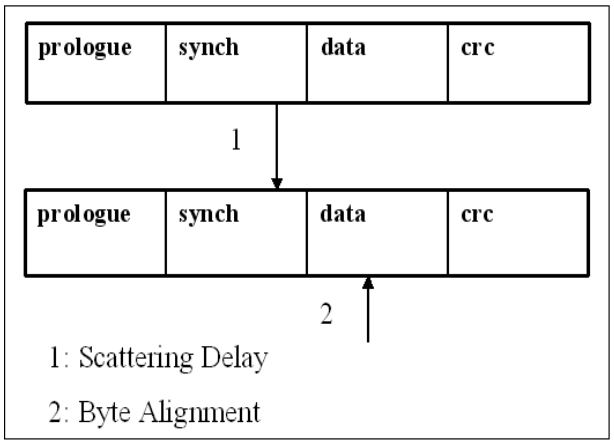

Figure1. Data packets transmitted over the radio channel

On the Mica2 [34] platform, the interrupt handling time is typically around 5μs depending on the length of the code path between the start of interrupt handler and the part that records the local time. However, we observed interrupt handling times as high as 30μs. The sum of encoding and decoding times is between 110μs and 112μs. The byte alignment time is between 0μs (for bit offset 0) and 365μs (for bit offset 7). In contrast, the propagation time is under 1μs. Table 1 summarizes the magnitudes and distribution of the various delays DATS in message transmissions.

The accuracy provided by the time synchronization service depends on the node and network resources available to the service. On the network side, accuracy increases as the frequency of the periodic message exchange increases. On the node side, accuracy can be improved using skew calculations.

In the basic version of the protocol, whenever a node receives a higher time value than its own, it copies the received value. However, this is inaccurate because the time value at the sender of that time has already moved forward by the time the value is copied. The amount by which its clock has moved forward is the sum of the time taken by the node to send out the message after time stamping it, the message transmission time and the time spent in receiving the message, comparing its time value and copying it.

To complicate matters, this elapsed time may be non-deterministic in case of random waiting or back-off strategies for channel access. The accuracy of the time synchronization service can be greatly improved by estimating this elapsed time. This calculation requires the sending of additional information in each message and additional processing at each node. In order to avoid errors due to non-deterministic delays, the stamping and copying of time values can be done at the MAC level just before the message is transmitted and just after it is received.

TABLE1. The sources of delays in message transmissions (DATS)

Time	Magnitude
Send and Receive	0-75 ms
Access	75-300 ms
Transmission	7-18 ms
Propagation	<1μs for distance up to 380 meters
Byte Alignment	0-480 ms

5. Evulation and Simulation

Our experiments demonstrate that the basic time synchronization algorithm meets the level of accuracy desired by the application, as can be seen from Table 2.

Our results also show that accuracy improves significantly when time synchronization is implemented as close to the hardware level as possible. Moreover, accuracy can be further improved using skew compensation techniques. It should be noted that in large scale networks, where the amount of message traffic received is high, processing time synchronization values at the level closest to hardware can be risky. Because of the overhead of extra computations at the lowest level, processing of time synchronization messages might be preempted by other low level events resulting in arbitrary state corruptions if not programmed or scheduled carefully.

With the purpose of simulate the proposed protocol, the acquired results are compared to the results from protocols [1], [12]. In comparison all the conditions involved are supposed to be the some, in the way that in all of them 50 sensor nodes are used. Result as shown figure 2.

TABLE2. Accuracy of the time synchronization protocol

Time Synch Schema	Per-Hop Accuracy
Basic Alg. (Local max. based)	305.1
Basic Alg. + low-level processing	19.25
Basic Alg. + low-level Processing + skew compensation	10.9

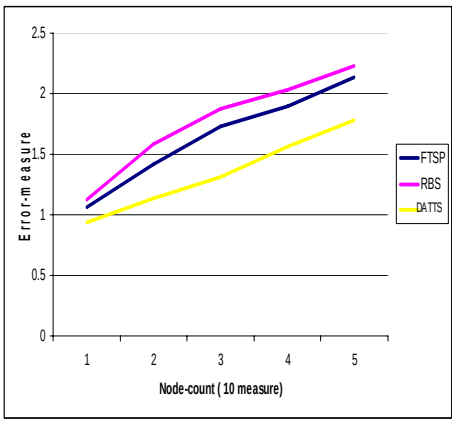

Figure2. Exist error in network and compare FTSP, RBS and proposed approach (DATS)

6. Conclusions

In this paper an algorithm to decrease transmission delay in wireless sensor network was introduced and discussed. We have described the Distributed algorithm for time synchronization in wireless sensor network. The protocol was implemented on the MATHLAB software. The precision of 1.78μs in the single hop scenario and the average precision of 0.94μs per hop in the multi-hop case were shown by providing experimental results. This performance is markedly better than those of other existing time synchronization approaches on the same platform.

The DATS was tested and its performance was verified in a real world application. This is significant because the service had to operate not in isolation, but as part of a complex application where resource constraints as well as intended and unintended interactions between components can and usually do cause undesirable effects. Moreover, the system operated in the field for extended periods and not under laboratory conditions. This is a testimony to the robustness of the protocol and its implementation.

The application, in addition to the DATS, contained several services, such as message routing, data aggregation, remote configuration and debugging services, along with application specific software components. A typical test scenario involved 50 to 60 motes distributed in an urban environment. The network was approximately 8 hops wide. The system was tested repeatedly for 4 to 8 hours of continuous operation. During testing some of the motes were switched off and on, the temperature and humidity of the environment changed drastically influencing the stability of the crystals. All nodes remained synchronized during these tests, but no other explicit time synchronization data was obtained. However, the overall performance of the counter sniper system and the fact that there was no performance degradation over time, clearly verified that the DATS performed well.

7. References

[1] Elson, J. E., Girod, L., and Estrin, D. Fine-Grained Network Time Synchronization using Reference Broadcasts. The Fifth Symposium on Operating Systems Design and Implementation (OSDI), p. 147–163, December 2002.

[2] W.R. Hein Zelman, A.Chandrakasan, and H. Balakrishnan. Energy efficient communication protocol for wireless networks. In proceedings of the 33rd System Sciences Hawaii International Conference on 4-7 2004 Page(s):10 pp. vol.2.

[3] A. Cemua, J. jure,, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In Proceedings of the 2006 ACM SIGCOMM USA, April 2006.

[4] Ganeriwal, S., Kumar, R., and Srivastava, M. B. Timing-Sync Protocol for Sensor Networks. The First ACM Conference on Embedded Networked Sensor System (SenSys), p. 138– 149, November 2003.

[5] L. Gemua. Development of an acoustic rangefinder. Technical Report 00-728, University of Southern California, Department of Computer Science, March 2004.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, pages 56–67, Boston, MA, Aug. 2005.

[7] Mills, D. L. Internet Time Synchronization: The Network Time Protocol. IEEE Transactions on Communications COM 39 no. 10, p. 1482–1493, October 2001.

[8] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli. Blind beam forming on a randomly distributed sensor array system. IEEE Journal of Selected Areas in Communications, 16(8):1555–1567, Oct 2004.

[9] Kay Romer. \Time Synchronization in Ad Hoc Networks." In Proceedings of MobiHoc 2001, Long Beach, CA, Oct 2001.

[10] Weilian Su and Ian F. Akyildiz. \Time-Diffusion Synchronization Protocol for Sensor Networks." Technical report, Georgia Institute of Technology, Broadband and Wireless Networking Laboratory, 2002.

[11] John R. Vig. \Introduction to Quartz Frequency Standards." Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate, October 2006. Available at http://www.ieeeu_c.org/ freqcontrol/quartz/vig/vigtc.htm.

[12] Mikos Maoti , Branislav Kusy , Gyula Simon and Akos ledeczi The Flooding Time Synchronization Protocol , Technical Institute for Software Integrated Systems, Vanderbilt University , 2005