Analysis and Comparative Study of Clock Synchronization Schemes in Wireless Sensor Networks

Surendra Rahamatkar
Associate Professor, Computer Science
Venkateshwar Institute of Technology
Indore, India

Dr. Ajay Agarwal
Professor, Computer Applications
Krishna Institute of Engg. & Technology
Ghaziabad, India

Narendra Kumar
Sr. Lecturer, Computer Science
Galgotia college of Engineering & Technology
Greater Noida, India

***Abstract*— Time synchronization is an important issue in wireless sensor networks. Many applications based on these WSNs assume local clocks at each sensor node that need to be synchronized to a common notion of time. Some intrinsic properties of sensor networks such as limited resources of energy, storage, computation, and bandwidth, combined with potentially high density of nodes make traditional synchronization methods unsuitable for these networks. Hence there has been an increasing research focus on designing synchronization schemes. This paper contains a survey, relative study and analysis of existing clock synchronization protocols for wireless sensor networks, based on a various factors that include precision, accuracy, cost, and complexity. The design considerations presented in this paper will help the designer in structuring a successful clock synchronization system. Specifically, the comparisons presented based on various factors will provide basic guidelines to the designer in integrating various solution features to create an efficient clock synchronization scheme for the application.**

Keywords- Wireless sensor networks, clock synchronization, protocol, comparison

I. 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. There are many areas 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. To achieve this 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 need 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.

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.

II. EXISTING APPROACHES TO TIME SYNCHRONIZATION

Time synchronization in sensor networks has attracted attention in last few years. *Post-facto synchronization* was a pioneering work by Elson and Estrin [13]. They proposed that unlike in traditional synchronization schemes such as NTP, local clocks of the sensor nodes should normally run unsynchronized in their own pace, but should synchronize whenever necessary. This way local timestamps of two nodes at the occurrence time of an event are synchronized later by extrapolating backwards to estimate the offset between clocks at a previous time (at the time of the event). In this section, building blocks and fundamental mechanisms of existing time synchronization algorithms are presented.

A. REFERENCE BROADCAST SYNCHRONIZATION (RBS)

RBS [7] provides synchronization for a whole network. The basic synchronization primitive is a reference broadcast to a set of client nodes in the one-hop neighborhood of a beacon node. The beacon node broadcasts manage synchronization pulses. The clients then exchange their respective reception times and use linear regression to compute relative offsets and rate differences to each other. Using offset and rate difference, each client can transform a local clock reading to the local timescale of any other client. To extend this scheme to multi-hop networks, the network is clustered such that a single beacon can synchronize all nodes in its cluster. Gateway nodes that participate in two or more clusters independently take part in the reference-broadcast procedure of all their clusters. By knowing offsets and rate differences to nodes in all adjacent clusters, gateway nodes can transform time stamps from one cluster to another. Time synchronization across multiple hops is then provided as follows. Nodes send their time-stamp data using their local clocks. Whenever time stamps are exchanged among nodes, the time stamps are transformed to the receiver's local time using offset and rate difference. In experiments it has been shown that adjacent Berkeley Motes can be synchronized with an average error of 11μs by using 30 broadcasts. Over multiple hops, the average error grows with $O(\sqrt{n})$, where n is the number of hops.

B. TIMING SYNCHRONIZATION PROTOCOL FOR SENSOR NETWORKS (TPSN)

TPSN [8] provides synchronization for a whole network. First, a node is elected as a synchronization master and a spanning tree with the master at the root is constructed by flooding the network. In a second phase, nodes synchronize to their parent in the tree by means of round-trip synchronization.

Synchronization is performed in rounds and initiated by the root and broadcasting a synchronization-request message to its children. Each child then performs a round-trip measurement to synchronize with the root. Nodes further down in the tree overhear the messages of their parents and start synchronization when their parents have synchronized. To eliminate message-delay uncertainties, time-stamping for the round-trip synchronization is done in the MAC layer. In case of node failures and topology changes, master election and tree construction must be repeated.

Measurements showed that two adjacent Berkeley Motes can be synchronized with an average error of 16.9μs, which is a worse figure than the 11μs given for RBS in [8]. However, the authors of [10] claim that a re-implementation of RBS on their hardware resulted in an average error of 29.1μs between adjacent nodes, effectively claiming that TPSN is about twice as precise as RBS.

C. TINYSYNC AND MINISYNC (TS/MS)

Tiny-Sync and Mini-Sync [9] are methods for pair wise synchronization of sensor nodes. Both Tiny-Sync and Mini-Sync use multiple round-trip measurements and a line-fitting technique to obtain the offset and rate difference of the two nodes. For this, a constant-rate model (see clock model) is assumed. To obtain data points for line fitting, multiple round-trip synchronizations are performed

Note that each of the two lines is unambiguously defined by two (a priori unknown) data points. The same results would be obtained if the remaining data points could be eliminated. Since the computational and memory overhead depends on the number of data points, it is a good idea to remove as many data points as possible before the line fitting. Tiny-Sync and Mini-Sync only differ in this elimination step. Essentially, Tiny-Sync uses a heuristic to keep only two data points

for each of the two lines. However, the selected points may not be the optimal ones. Mini-Sync uses a more complex approach to eliminate exactly those points that do not change the solution. Hence, Tiny-Sync achieves a slightly suboptimal solution with minimal overhead, Mini-Sync gives an optimal solution with increased overhead.

Measurements on a 802.11b network with 5000 data points resulted in an offset bound of 945μs (3230μs) and a rate bound of 0.27ppm (1.1ppm) for adjacent nodes (nodes five hops away).

D. Time Diffusion Synchronization (TDP)

TDP [10] supports the synchronization of a whole network. Initially, a set of master nodes is elected. For external synchronization, these nodes must have access to a global time. This is not required for internal synchronization, where masters are initially unsynchronized.

Master nodes then broadcast a request message containing their current time, and all receivers send back a reply message. Using these round-trip measurements, a master node calculates and broadcasts the average message delay and standard deviation.

Receiving nodes record these data for all leaders. Then, they turn themselves into so-called "diffused leaders" and repeat the procedure. The average delays and standard deviations are summed up along the path from the masters. The diffusion procedure stops at a given number of hops from the masters. In a simulation with 200 static nodes with 802.11 radios and a delay of 5 seconds between consecutive synchronization rounds, the deviation of time across the network dropped to 0.6 seconds after about 200 seconds.

E. Asynchronous Diffusion (AD)

AD [11] supports the internal synchronization of a whole network. The algorithm is very simple: each node periodically sends a broadcast message to its neighbors, which reply with a message containing their current time. The receiver averages the received time stamps and broadcasts the average to the neighbors, which adopt this value as their new time. It is assumed that this sequence of operations is atomic, that is the averaging operations of the nodes must be properly sequenced. In a simulation with a random network of 200 static nodes showed that the synchronization error decreases exponentially with the number of rounds.

III. Evaluation and Comparison of Protocols

In this section we compare and evaluate the various synchronization protocols. Before evaluating the various protocols, we define the criterion in detail first, which is used in comparisons. For this, we have evaluated the presented protocols in two ways (1) quantitative criteria and (2) qualitative criteria. This contains synchronization accuracy, computational complexity, and convergence time. A qualitative criterion includes scalability, energy efficiency, and fault-tolerance. By combining together, these factors provide a good characterization of the applicability and performance of each protocol.

A. Quantitative Evaluation

Here, the protocols differ broadly in their computational requirements, energy consumption, precision of synchronization results, and communication requirements is presented. In addition, no protocol clearly outperforms the others in all possible applications of wireless networks. Rather, it is quite likely that the choice of a protocol will be driven by the characteristics and requirements of each application. For instance, a low-cost, low-precision protocol could be appropriate for many environmental monitoring applications. However, many safety-critical applications, such as aircraft navigation or intrusion detection in military systems, will demand high-precision protocols in order for nodes to correctly identify events occurring in the net and for an application to respond to those events.

Synchronization precision: Each network node has a physical clock consisting of hardware oscillator circuits. Unfortunately, the frequency of hardware clocks varies from one node to another within a specified range. Thus, clocks on different nodes in wireless networks operate at different rates. Consequently, the clock values used for synchronization in wireless networks are not physical clock readings. Instead, network nodes generally use a logical notion of clocks and time. Logical clocks can be modified both by software (e.g., during synchronization) and hardware (e.g., by the physical clocks). Consequently, synchronization precision can be defined in two ways.

1. Absolute precision: The maximum error (i.e., skew and offset) of a node's logical clock with respect to an external standard such as UTC.

2. Relative precision: The maximum deviation (i.e., skew and offset) among logical clock readings of the nodes belonging to a wireless network.

In general, high synchronization precision is clearly a desirable feature of a synchronization protocol. However, in the protocols that we studied, higher synchronization precision comes at the expense of increased computational cost measured in terms of algorithmic complexity, the number of messages exchanged among nodes, and the storage requirements of the protocol. The quantitative precision of the various protocols appears in Table 1.

Piggybacking: Piggybacking is the process of combining the acknowledgement messages during

synchronization with messages that carry synchronization data among nodes. Instead of sending independent acknowledgement messages, these messages are piggybacked on the data messages that have to be sent to the node, in order to reduce message traffic in the network. Piggybacking is clearly advantageous because wireless networks are often subject to severe bandwidth constraints and piggybacking alleviates communication demands on the network. In addition, piggybacking can also reduce the storage requirements on network nodes because storage space is also saved by clubbing acknowledgements with data messages. TPSN time synchronization [8] use piggybacking.

Computational complexity: As wireless networks often have limited hardware capabilities and severe energy constraints, the complexity of a synchronization protocol can make a protocol impractical for many applications. Here we distinguish between the computational complexity of a protocol (i.e., its run-time and memory requirements) from the message complexity (i.e., the number of messages exchanged during synchronization). (See also discussion on convergence time below.)

In our evaluations, we consider both the asymptotic behavior of a protocol's computation time and its memory requirements, relative to the number of nodes being synchronized. Even though a protocol's computational requirements might be linear in the number of nodes synchronizing with each other, the protocol may still be impractical if these requirements exceed physical node resources. RBS [7], asynchronous diffusion [11], and TDP[10] have higher computational and storage complexity compared to TPSN [8], and the Tiny Sync and Mini Sync[9].

Convergence time: Convergence time is the total time required to synchronize a network. A protocol that requires a large number of message exchanges per synchronization will result in a longer convergence time. RBS [7] does not emphasize low convergence time because they are based on reactive routing. In these protocols, synchronization is performed relatively infrequently, when an event of interest occurs in the net. Thus, convergence time and message complexity are of less critical importance in protocols that use reactive routing. The protocols Tiny Sync and Mini Sync [9]can tolerate high convergence times for multi-hop networks.

Network size: Some authors have conducted empirical evaluations of synchronization protocols on actual sensor networks. Although this information is not available for most of the protocols that we studied, the TPSN time synchronization protocol of Ganeriwal et al. [8] is noteworthy in this regard. This protocol was found to handle neighborhoods with up to 300 nodes.

Compatibility with sleep mode: The ability of a node to be in low-power (sleep) mode can be critical to meeting the node's energy requirements. The key idea underlying *sleep mode* is that nodes must be synchronized and active only when the application demands it. RBS [7] highlights this feature by way of post-facto synchronization and other protocols TPSN [8], Tiny[9] support this feature as well.

TABLE I. QUANTITATIVE PERFORMANCE COMPARISON OF SYNCHRONIZATION PROTOCOLS

Protocols	Precision	Piggy backing	Complexity	Conver -gence time	Network size	Sleep mode
RBS [7]	1.85 ± 1.28 µs	N/A	High	N/A	2-20 Nodes	Yes
TPSN [8]	16.9 µs	No	Low	Unkno wn	150-300 Nodes	Yes
TS/MS[9]	945 µs	No	Low	High (Multi-hop)	N/A	Yes
TDP [10]	100 µs	No	High	High (Multi-hop)	200 Nodes	Yes
AD [11]	Unkno wn	No	High	High (Multi-hop)	200–400 Nodes	Yes

TPSN time synchronization [8] is an excellent compromise among synchronization accuracy, computational complexity, and convergence time. While the accuracy results are in the order of tens of microseconds, low computational complexity and fast convergence time make this protocol quite attractive when higher accuracy is not required. An additional strength of this protocol is that the protocol has been tested on an actual sensor network containing 300 nodes. The delay-measurement time synchronization protocol has comparable accuracy results as network-wide time synchronization [8]. However, this result is obtained at the expense of a longer convergence time. Improvements to TPSN synchronization were defined by Dai and Han [12]. Their method has yielded excellent accuracy results with lower message complexity than RBS. Finally, a protocol by Sichitiu and Veerarittiphan [9] achieves good accuracy results (less than one millisecond). This is quite impressive considering that the protocol also has low computational complexity. However, the convergence time of this protocol is quite high. The asynchronous diffusion protocol of Li and Rus [11] and the time diffusion protocol of Su and Akyildiz [10] use an averaging method to adjust node clocks. These protocols are reasonably robust; however, the issue of clocks running backward must be suitably addressed before these protocols are implemented in practice.

B. QUALITATIVE EVALUATION

In this section we evaluate the protocols based on overall quality criteria. In contrast to Section 3.1 above,

here the goals of each protocol are discussed and the extent to which we consider that the protocol succeeds in achieving those goals. While a quantitative study deals with parameters that help the reader fine-tune a synchronization protocol by providing a telescopic view, a qualitative study provides a broader and more general perspective. Table 2 compares the various protocols in terms of the following qualitative criteria.

Energy efficiency: Energy efficiency is an implicit requirement in most wireless networks. The extent to which this requirement must be enforced will vary depending on an application. For instance, in the case of sensor networks the requirement is quite strict, forcing nodes to sleep as frequently as possible and severely limiting the energy available for synchronization and other network tasks. The main reason behind this energy constraint is the small size of batteries in sensor nodes, which greatly limits the amount of energy that can be stored and produced (e.g., with solar cells). An important tradeoff for wireless networks is between using the available energy for computing or for communicating.

Accuracy: Accuracy is a measure of how well the time maintained within the network is true to the standard time. In other words, it is a measure of the precision of synchronization. A protocol with high accuracy thereby guarantees high precision. In the case of absolute precision, this means that the synchronized time in the network does not deviate much from an external standard (e.g., UTC or GPS). In the case of relative precision, this means that, when a set of synchronized nodes is considered, the maximum deviation of the clock of any node within the set is reasonably small.

Scalability: Scalability is the scope of a network that is the geographic span of nodes that are synchronized and the completeness of coverage within that region. In general, the scope of a network can be expanded by increasing the number of nodes in the network. As the sensors are becoming cheaper, wireless sensor networks are becoming increasingly large, up to tens of thousands of nodes. Thus, synchronization protocols must be sufficiently scalable with respect to network size. Most protocols that handle sensor networks place scalability on top of their list of priorities.

Although most authors have not measured scalability in their experiments, Ganeriwal et al. [8] have used a network of 150-300 nodes to test scalability. In addition, Su and Akyildiz used a net with 200 nodes to test the scalability of TDP [10]. RBS [7] typically synchronizes 3-20 nodes in a neighborhood; however, this protocol works well even in much larger networks using gateways between neighborhoods.

Overall complexity: The quantitative evaluation in the previous subsection distinguishes various complexity measures, including CPU load, storage requirements, and message complexity (i.e., convergence time). In this section, overall complexity is viewed as a combination of algorithmic complexity, overhead caused due to fault tolerance provisions, and communication overhead.

Fault tolerance: Fault tolerance plays an important role because a wireless medium is rather error-prone. The poor reliability of message delivery in a wireless medium can have devastating effects on synchronization protocols because synchronization requires message exchanges.

Some fault-tolerant protocols [8, 10] address message loss to some extent, but others have not addressed this issue. Consequently, it is unclear how sensitive their protocols are to message loss. This is somewhat troublesome because handling message loss can result in significant overheads and performance degradation during synchronization.

TABLE II: QUALITATIVE PERFORMANCE COMPARISONS OF SYNCHRONIZATION PROTOCOLS.

Protocols	Accuracy	Energy Efficiency	Overall Complexity	Scalability	Fault Tolerance
RBS [7]	High	High	High	Good	No
TPSN [8]	High	Average	Low	Good	Yes
TS/MS [9]	High	High	Low	N/A	Yes
TDP [10]	High	Average	High	Good	Yes
AD [11]	Unknown	Low	High	N/A	Yes

IV. CONCLUSIONS

Wireless sensor networks have tremendous advantages for monitoring object movement and environmental properties but require some degree of synchronization to achieve the best results. With increasing frequency, attention has been focused on wireless sensor networks because of their wide range of application areas. Among the many difficulties in designing and building such sensor networks, a essential challenge is providing clock synchronization among the sensor nodes.

In wireless sensor network traditional clock synchronization protocols for wired networks cannot be used because the sensor protocols deals with dynamical behavior, the ability to handle sensor mobility, and scalability. Due to limited energy resources the sensors themselves are heavily resource-constrained. Therefore, they need to operate in highly unreliable environments.

In this paper we presented a survey, relative study and analysis of existing clock synchronization protocols for wireless sensor networks, based on a range of factors that includes precision, accuracy, cost, and complexity.

The design considerations presented in this paper will help the designer in structuring a successful clock synchronization system.

Specifically, the comparisons presented based on various factors will provide basic guidelines to the designer in integrating various solution features to create a successful clock synchronization scheme for the application.

REFERENCES

[1] I.-K Rhee, J Lee, J. Kim, E. Serpedin, Wu, Y.-C. "Clock Synchronization in Wireless Sensor Networks: An Overview". Sensors, 2009: pp 56-85.

[2] F. Zhao, L.Guibas, "Wireless Sensor Networks: An Information Processing Approach. Morgan Kaufmann:" San Francisco, CA, USA: 2004, pp 107-117.

[3] D. L. Mills, "Internet time synchronization: the network time protocol." IEEE Trans. Commun. , 1991, 10: pp 1482-1493.

[4] N. Bulusu,; S. Jha, "Wireless Sensor Networks: A Systems Perspective", Artech House: Norwood MA, USA, 2005.

[5] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System", Communications of the ACM 21, 1978, pp 558-565

[6] T.K. Srikanth, , S. Toueg, "Optimal Clock Synchronization" Journal of the ACM 34, 1987, pp 626-645.

[7] Elson Jeremy, Girod Lewis, and Estrin Deborah, "Fine-grained network time synchronization using reference broadcasts" In Fifth Symposium on Operating Systems Design and Implementation OSDI, 2002.

[8] Saurabh Ganeriwal, Kumar Ram, and Mani B. Srivastava "Timing-sync protocol for sensor networks". In First ACM Conference on Embedded Networked SensorSystems (SenSys, 2003.

[9] L. Sichitiu Mihail, Chanchai Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks", In IEEE Wireless Communications and Networking Conference (WCNC), 2003.

[10] I. W. Su Akyildiz, "Time-Diffusion Synchronization Protocols for Sensor Networks", IEEE/ACM Transactions on Networking, 2005.

[11] Li Qun and Rus Daniela, " Global clock synchronization in sensor networks"' In IEEE InfoCom, 2004.

[12] H. Dai and R. Han, "TSync A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks", ACM SIGMOBILE Mobile Computing and Communications Review, 8(1) 2004 , pp 125-139.

[13] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks", International Parallel and Distributed Processing Symposium, San Francisco, USA, 2001.

AUTHORS PROFILE

Surendra Rahamatkar has received his Bachelor of Engineering in Computer Science & Engineering from Barkatullah University, Bhopal, India, Post Graduate Diploma in Advanced Computing from CDAC, Pune and Master in Technology in Computer Science & Engineering from VMRF Deemed University, India. He is working as Associate Professor in the Department of Computer Science & Engineering at Venketeshwar Institute of Technology, Indore, India. He is a member of various Technical Societies viz. Computer Society of India (CSI), International Association of Engineers (IEA), Indian Society of Technical Education (ISTE). He published many research papers in various International/ National Journals and Conferences. He is presently working on Time Synchronization in Wireless Sensor Networks towards his Ph.D. degree. His main research interests include: Wireless Sensor Network, Distributed & Mobile Computing and Middleware.

Dr. Ajay Agarwal has done B.Tech. Degree in Computer Science & Engineering from Institute of Engineering & Technology, Lucknow (India), M.Tech.(honors) Degree in Computer Science & Engineering from Motilal Nehru Regional Engineering College, Allahabad and Ph.D. in Computer Science from Indian Institute of Technology, Delhi (India). Presently he is working as a Professor and Head in Computer Application Department at Krishna Institute of Engineering & Technology, Ghaziabad, India. He is a member of various Technical Societies viz. Institute of Electrical and Electronics Engineers (IEEE), Computer Society of India (CSI), Indian Society for Technical Education (ISTE), Institution of Engineers India, Institute of Chartered Computer Professional of India and Indian Association of Physics Teachers. He published many papers in various International/ National Journals and Conferences. His main research interests include: Wireless Sensor Network, Mobile Computing and Middleware.

Narendra Kumar has received his Bachelor of Technology and Master in Technology in Computer Science & Engineering from UP Technical University, Lucknow, India. He is working as Senior Lecturer in the Department of Computer Science & Engineering at Galgotia College of Science & Technology, Greater Noida, India. He is a member of various Technical Societies viz. Computer Society of India (CSI), Indian Society of Technical Education (ISTE). He published many research papers in various Conferences. His main research interests include: Wireless Sensor Network, Distributed & Mobile Computing and Middleware.