Scientific Journal Impact Factor (SJIF): 1.711

e-ISSN: 2349-9745
p-ISSN: 2393-8161

International Journal of Modern Trends in Engineering and Research
www.ijmter.com

Distributed Approach for Clock Synchronization in Wireless Sensor Network

Ankit K. Patel[1], Ankur Chauhan[2]
[1]*Assistant Professor, Computer Engineering, Ahmedabad Institute of Technology, Gujarat, India.*
[2]*Assistant Professor, Information Technology, Ahmedabad Institute of Technology, Gujarat, India.*

***Abstract*—** Time synchronization is an important service in WSNs. existing time synchronization algorithms provide on average good synchronization between arbitrary nodes, however, as we show in this paper, close-by nodes in a network may be synchronized poorly. We propose the Distributed Time Synchronization Algorithm (DTSA) which is designed to provide accurately synchronized clocks between nearest-neighbours. DTSA works in a completely decentralized fashion: Every node periodically broadcasts its time information. Synchronization messages received from direct neighbours are used to calibrate the logical clock. The algorithm requires neither a tree topology nor a reference node, which makes it robust against link and node failures.

Keywords — mote, global clock, MAC, WSN

I. INTRODUCTION

Recent advances in micro-electromechanical (MEMS) technology have led to the development of small, low-cost, and low-power sensors. Wireless sensor networks (WSNs) are large-scale networks of such sensors, dedicated to observing and monitoring various aspects of the physical world. In such networks, data from each sensor is agglomerated using data fusion to form a single meaningful result, which makes time synchronization between sensors highly desirable. This dissertation work surveys and evaluates existing clock synchronization protocols based on a palette of factors like precision, accuracy, cost, and complexity and defining a new protocol that is best suited to the specific needs of a sensor-network application that is DTSA (Distributed Time Synchronization Algorithm). Finally, the work provides a

Valuable framework by which designers can compare new and existing synchronization protocols. In addition, time synchronization is significant as sensor Network protocols make use of time in various forms. Media access control using TDMA needs accurate time information, so that transmissions do not interfere. Similarly, to save energy, sensor network protocols often employ advanced duty-cycling schemes, and turn off their radio if not needed [3]. An accurate time helps to save energy by shortening the necessary wake-up guard times. Although each sensor node is equipped with a hardware clock, these hardware clocks can usually not be used directly, as they suffer from severe drift. No matter how well these hardware clocks will be calibrated at deployment, the clocks will ultimately exhibit a large skew. To allow for an accurate common time, nodes need to exchange messages from time to time, constantly adjusting their clock values.

Although multi-hop clock synchronization has been studied extensively in the last decade, we believe that there are still facets which are not understood well, and eventually need to be addressed. One such issue is *locality*: Naturally, one objective in clock synchronization is to minimize the skew between any two nodes in the network, regardless of the "distance" between them. This is known as *global* clock skew minimization. Indisputable, having two far-away nodes well-synchronized is a noble goal, but is it really what we require most? In this paper, we argue that accurate clock synchronization between neighboring nodes is often at least as important. In fact, all examples mentioned earlier tolerate suboptimal global clock synchronization: Guessing the location of a

commonly sensed acoustic signal needs precise clock synchronization between all the nodes that are able to sense the signal. Similarly, in a MAC layer that is optimized for throughput or energy, we care that possibly interfering (neighboring) nodes have a precise clock. In contrast, global skew is not of great concern; it is perfectly tolerable if far-away nodes have larger pair-wise error. This is known as *local* clock skew minimization. Optimally, we would like to have a clock synchronization protocol that is precise in the direct neighborhood, and maybe a bit less so in the extended neighborhood.

Current state-of-the-art multi-hop clock synchronization protocols such as FTSP [4] are designed to optimize the global skew. However, as we will show in this paper, there is room for improvement regarding the local skew. This is not really surprising, as FTSP and similar protocols work on a spanning tree, synchronizing nodes in the tree with their parents, and ultimately with the root of the tree. Neighboring nodes which are not closely related in the tree, i.e., where the closest common ancestor even is the root of the tree, will not be synchronized well because errors propagate down differently on different paths of the tree, see Figure 1.Eliminating all the deterministic sources of errors, the remaining two-hop error would be totally symmetric in the best case [5]. Indeed, every hop will experience some kind of inevitable random error δ. As randomly distributed errors sum up according to the square-root function on each hop, the expected error between head and tail of a chain of k nodes is in the order of $\delta \sqrt{k}$. Therefore, two nodes that are not in the same subtree rooted at the reference node are expected to experience an error in the order of the square-root of their distance in the tree.

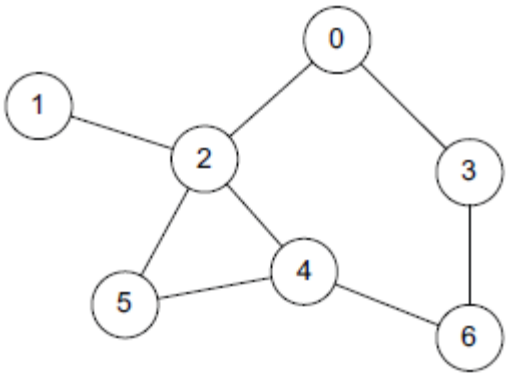


Figure-1

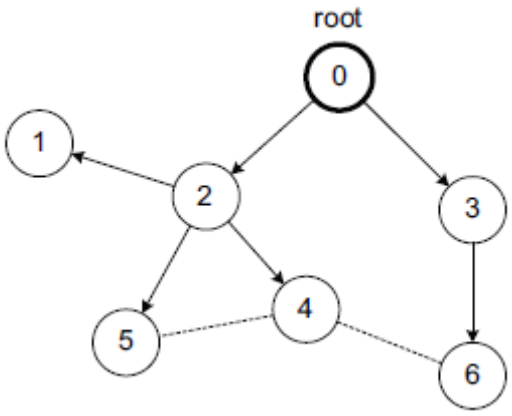


Figure-2

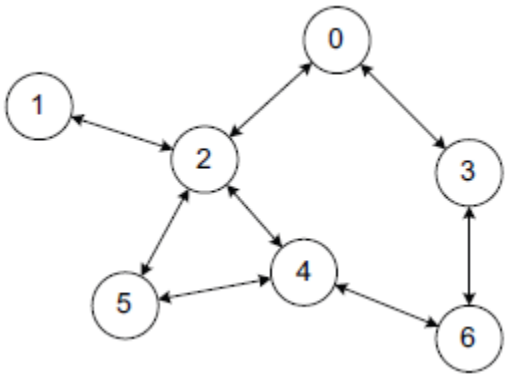


Figure-3

In Figure-1, we see a typical sensor network; edges between sensor nodes indicate a bidirectional communication link. Figure-2 represents a tree-based synchronization protocol with node 0 as reference clock (root), where every node in the tree synchronizes with its parent; in this example we would expect nodes 4 and 6 to synchronize suboptimally even though they are direct neighbors, because they are part of different subtrees. Finally, In Figure-3 we see the idea of Distributed Time synchronization Algorithm (DTSA): Every node synchronizes with all its neighbors in the communication graph. No root node is necessary. Here each node performs averaging of time stamp of synchronization message with its local time only if the time stamp of the received Synchronization message is grater then it's local clock.

In the theory community, clock synchronization has been studied for many years, recently with a focus on the local (also known as *gradient*) clock skew, e.g., [6, 7]. The goal of this paper is to investigate whether these theoretical insights carry over to practice. In particular, in Sections 4, we will propose the Distributed Time Synchronization Algorithm (DTSA), a clock synchronization protocol that excels primarily at local clock synchronization. It is inspired by a long list of theoretical papers, originating in the distributed computing community [8, 9, 10, 11], lately also being adopted by the control theory community [12]. As such, DTSA is completely distributed, relying only on local information, requiring no reference node or tree construction. We argue that this approach results in a better average synchronization between neighbors while still maintaining a tolerable global skew.

II. RELATED WORK

Clearly, clock synchronization has been studied extensively, long before the advent of wireless sensor networks. The classic solution is an atomic clock, such as in the global positioning system (GPS). Equipping each sensor node with a GPS receiver is feasible, but there are limitations in the form of cost and energy. Moreover, line of sight to the GPS satellites is needed, limiting the use to outdoor applications.

Classical clock synchronization algorithms rely on the ability to exchange messages at a high rate which may not be possible in wireless sensor networks. Traditional time synchronization algorithms like the *Network Time Protocol (NTP)* [13] are due to their complexity not well suited for sensor network applications. Moreover, as their application domain is different, they are not accurate enough for our purpose, even in a LAN they may experience skew in the order of milliseconds. Sensor networks require sophisticated algorithms for clock synchronization since the hardware clocks in sensor nodes are often simple and may experience significant drift. Also, in contrast to wired networks, the multi-hop character of wireless sensor networks complicates the problem, as one cannot simply employ a standard client/server clock synchronization algorithm. As research in sensor networks evolved during the last years, many different approaches for time synchronization were proposed. Romer presents a system [14] where events are time-stamped with the local clock. When such a timestamp is passed to another node, it is converted to the local timestamp of the receiving node. *Reference Broadcast Synchronization (RBS)* [15] exploits the broadcast nature of the physical channel to synchronize a set of receivers with one another. A reference node is elected within each cluster to synchronize all other nodes.

Since differences in the propagation times can generally be neglected in sensor networks, a reference message arrives at the same instant at all receivers. The timestamp of the reception of a broadcast message is recorded at each node and exchanged with other nodes to calculate relative clock offsets. RBS is designed for single-hop time synchronization only. However, nodes which participate in more than one cluster can be employed to convert the timestamps between local clock values of different clusters. Pulses from an external clock source attached to one node, for example a GPS receiver, can be treated like reference broadcasts to transform the local timestamps into UTC.

The *Timing-sync Protocol for Sensor Networks (TPSN)* [16] aims to provide network-wide time synchronization. The TPSN algorithm elects a root node and builds a spanning tree of the network during the initial level discovery phase. In the synchronization phase of the algorithm, nodes synchronize to their parent in the tree by a two-way message exchange. Using the timestamps embedded in the synchronization messages, the child node is able to calculate the transmission delay and the relative clock offset. However, TPSN does not compensate for clock drift which makes frequent resynchronization mandatory. In addition, TPSN causes a high communication overhead since a two-way message exchange is required for each child node. These shortcomings are tackled by the *Flooding-Time Synchronization Protocol (FTSP)* [4]. A root node is elected which periodically floods its current timestamp into the network forming an ad-hoc tree structure. MAC layer time-stamping reduces possible sources of uncertainty in the message delay. Each node uses a linear regression table to convert between the local hardware clock and the clock of the reference node. The root node is dynamically elected by the network based on the smallest node identifier. After initialization, a node waits for a few rounds and listens for synchronization beacons from other nodes. Each node sufficiently synchronized to the root node starts broadcasting its estimation of the global clock. If a node does not receive Synchronization messages during a certain period, it will declare itself the new root node.

The *Routing Integrated Time Synchronization protocol (RITS)* [17] provides post-facto synchronization. Detected events are time-stamped with the local time and reported to the sink. When such an event timestamp is forwarded towards the sink node, it is converted from the local time of the sender to the receiver's local time at each hop. A skew compensation strategy improves the accuracy of this approach in larger networks. A completely distributed synchronization algorithm was proposed in [18]. The *Reachback Firefly Algorithm (RFA)* is inspired from the way neurons and fireflies spontaneously synchronize. Each node periodically generates a pulse (message) and observes pulses from other nodes to adjust its own firing phase. The authors report that a synchronization accuracy of 100μs can be achieved with this approach. RFA only provides synchronicity; nodes agree on the firing phases but do not have a common notion of time. Another shortcoming of RFA is the fact that it has a high communication overhead.

The fundamental problem of clock synchronization has been studied extensively and many theoretical results have been published which give bounds for the clock skew and communication costs [9, 11]. Srikanth and Toueg [10] presented a clock synchronization algorithm which minimizes the global skew, given the hardware clock drift.

The *gradient clock synchronization problem* was first introduced by Fan and Lynch in [6]. The gradient property of a clock synchronization algorithm requires that the clock skew between any two nodes is bounded by the distance (uncertainty in the message delay) between the two nodes. They prove a lower bound for the clock skew of $\Omega(d + \log D \log \log D)$ for two nodes with distance d, where D is the network diameter. This lower bound also holds if delay uncertainties are neglected and an adversary can decide when a sync message will be sent [19]. Recently, Lenzen et al. [20] proposed a distributed clock synchronization algorithm guaranteeing clock skew $O(\log D)$ between neighboring nodes while the global skew between any two nodes is bounded by $O(D)$.

III. SYSTEM MODEL

In this section, we introduce the system model used throughout the rest of this paper. We assume a network consisting of a number of nodes equipped with a hardware clock subject to clock drift. Furthermore, nodes can convert the current hardware clock reading into a logical clock value and vice versa.

A. Hardware Clock

Each sensor node i is equipped with a hardware clock $Hi(\cdot)$. The clock value at time t is defined as

$$H_i(t) = \int_{t_0}^{t} h_i(\tau)\, d\tau + \Phi_i(t_0)$$

Where $hi(\tau)$ is the hardware clock rate at time τ and $\Phi i(t0)$ is the hardware clock offset at time $t0$.
It is assumed that hardware clocks have bounded drift, i.e., there exists a constant $0 \leq \rho < 1$ such that
$1 - \rho \leq h(t) \leq 1 + \rho$
for all times t. This implies that the hardware clock never stops and always makes progress with at least a rate of $1 - \rho$. This is reasonable assumption since common sensor nodes are equipped with external crystal oscillators which are used as clock source for a counter register of the microcontroller. These oscillators exhibit drift which is only gradually changing depending on the environmental conditions such as ambient temperature or battery voltage and on oscillator aging. This allows assuming the oscillator drift to be relatively constant Over short time periods. Crystal oscillators used in sensor nodes normally exhibit a drift between 30 and 100 ppm.

B. Logical Clock

Since other hardware components may depend on a continuously running hardware clock, its value should not be adjusted manually. Instead, a logical clock value $Li(\cdot)$ is computed as a function of the current hardware clock. The logical clock value $Li(t)$ represents the synchronized time of node i. It is calculated as follows:

$$L_i(t) = \int_{t_0}^{t} h_i(\tau) \cdot l_i(\tau)\, d\tau + \theta_i(t_0)$$

where $li(\tau)$ is the *relative logical clock rate* and $\theta i(t0)$ is the clock offset between the hardware clock and the logical clock at the reference time $t0$. The logical clock is maintained as a software function and is only calculated on request based on a given hardware clock reading.

IV. SYNCHRONIZATION ALGORITHM

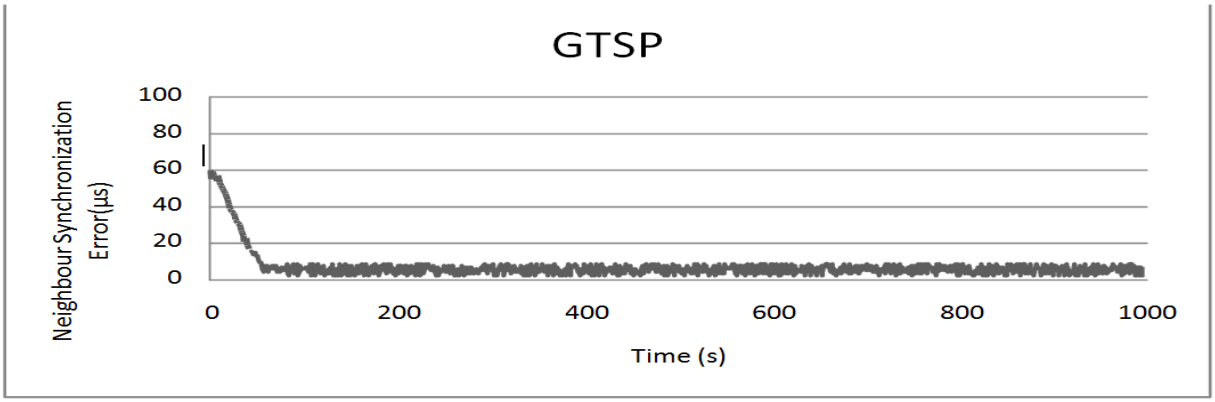
In this section, we describe our distributed clock synchronization algorithm. The basic idea of the algorithm is to provide precise clock synchronization between direct neighbors while each node can be more loosely synchronized with nodes more hops away.
In a network consisting of sensor nodes with perfectly calibrated clocks (no drift), time progresses at the same rate throughout the network. It remains to calculate once the relative offsets amongst the nodes, so that they agree on a common global time. However, real hardware clocks exhibit relative drift in the order of up to 100 ppm leading to a continually increasing synchronization error between nodes. Therefore, it is mandatory to repeat the synchronization process frequently to guarantee certain bounds for the synchronization error.
However, precisely synchronized clocks between two synchronization points can only be achieved if the relative clock drift between nodes is compensated. In structured clock synchronization algorithms all nodes adapt the rate of their logical clock to the hardware clock rate of the reference node.

Here we show the pseudo code of proposed algorithm.

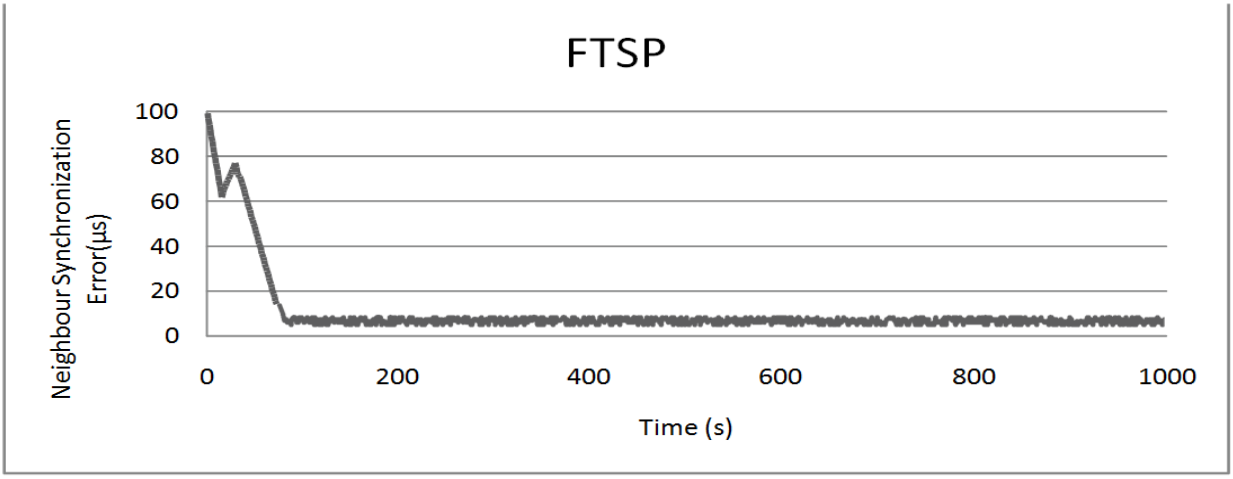
```
Event HendalMessage(TimeSyncMsg *msg)

{

If(msg->time_stamp <= Node_local_clock)

        {
                Delete message;
                Exit;
         }

else

        {
                avg=(msg->time_stamp                    +
Node_local_clock)/2;
               Node_local_clock=Node_SetTime(avg);
         }

}
```

Proposed clock synchronization algorithm is completely distributed and reliable to link and node failures, it is not practicable to synchronize to the clock of a reference node. Therefore, our clock synchronization algorithm strives to agree with its neighbors on the current logical time. Having synchronized clocks is a twofold approach; one has to agree both on a common logical clock rate and on the absolute value of the logical clock.

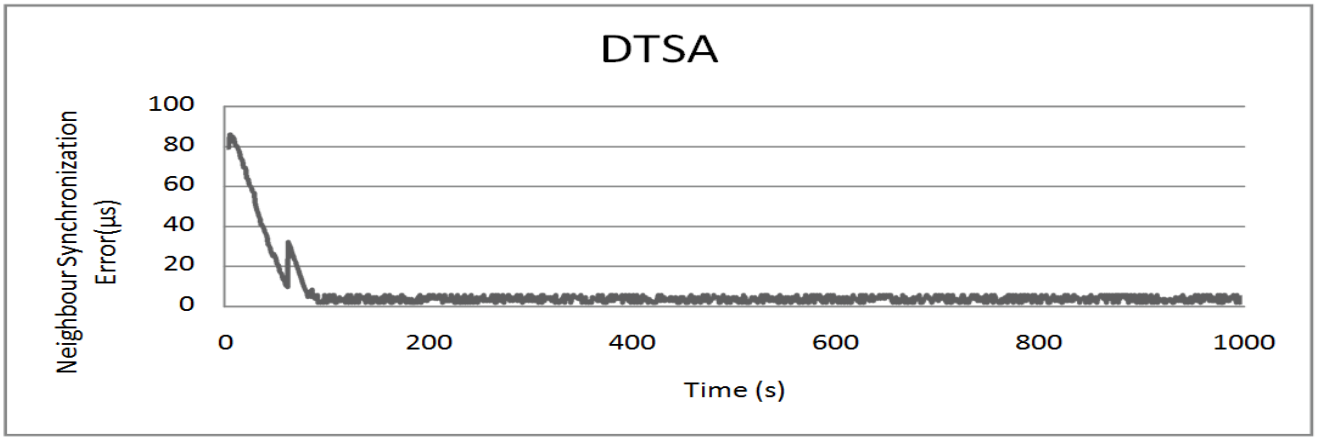
V. COMPARISON OF RESULTS



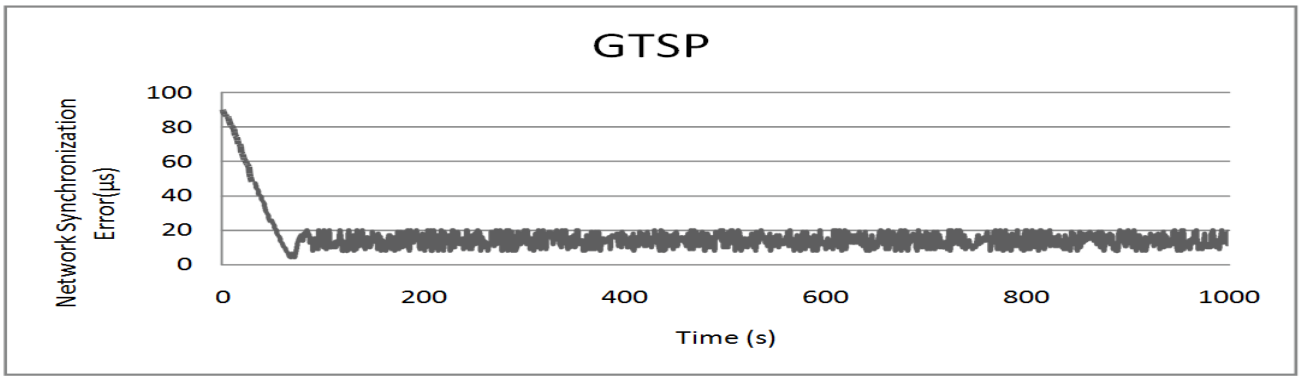
An Average Neighbor Synchronization Error (5.4µs) measured for GTSP (Topology-Random, Node-10).



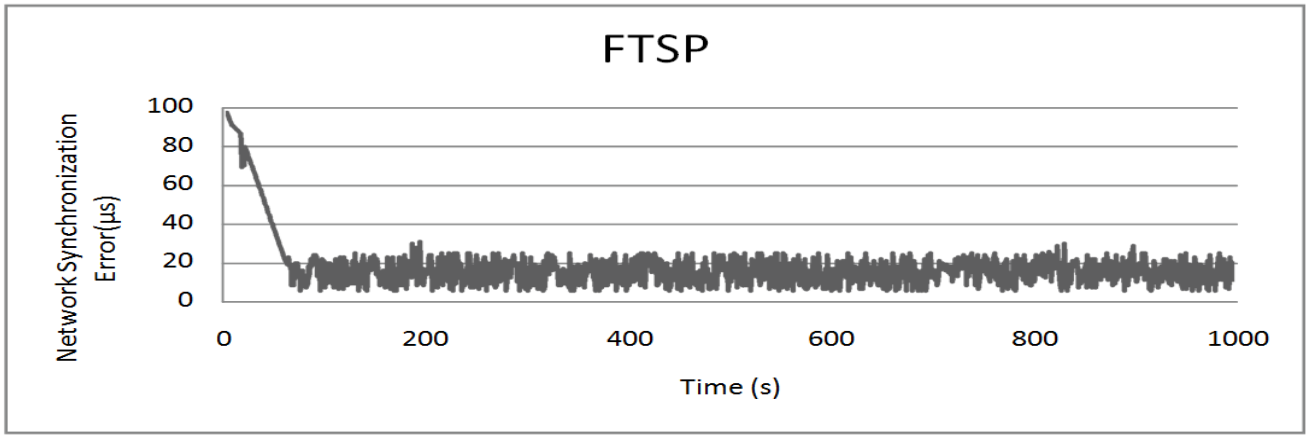
An Average Neighbor Synchronization Error (8.3µs) measured for FTSP(Topology-Random, Node-10).



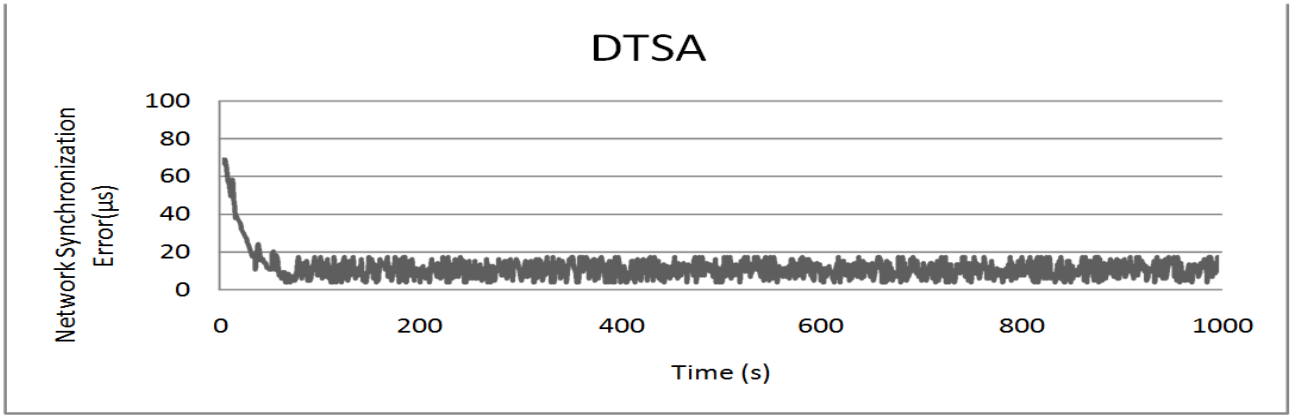
An Average Neighbor Synchronization Error (4.7µs) measured for DTSA(Topology-Random, Node-10).



An Average Network Synchronization Error (14.7µs) measured for GTSP (Topology-Random, Node-10).



An Average Network Synchronization Error (18.9µs) measured for FTSP (Topology-Random, Node-10).



An Average Network Synchronization Error (12.2µs) measured for DTSA (Topology-Random, Node-10).

VI. CONCLUSION AND FUTUREWORK

Sensor Clock Synchronization is a significant issue in Wireless Sensor network. In the presented work, I have made a study of GTSP and FTSP Time Synchronization Protocols. We presented the Distributed Time Synchronization Algorithm (DTSA) which is a completely distributed time synchronization protocol. Nodes periodically broadcast synchronization beacons to their neighbors. Using a simple update algorithm, they try to agree on a common logical clock with their neighbors. It can be shown by simulation analysis that by employing this algorithm, the DTSA provide more accuracy as compare to GTSP and FTSP. DTSA relies on local information only, making it robust to node failures and changes in the network topology.

In future we will study the performance of DTSA with respect to energy efficiency, and also try to compare the behavior of DTSA with GTSP and FTSP synchronization protocols for energy efficiency. I sincerely hope that my work will contribute in providing further research directions in the area of clock synchronization in wireless sensor network.

REFERENCES

[1] M. Allen, L. Girod, R. Newton, S. Madden, D. T. Blumstein, and D. Estrin. Vox Net: An Interactive, Rapidly-Deployable Acoustic Monitoring Platform. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, 2008.

[2] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based counter sniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[3] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low Power Data Gathering in Sensor Networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 2007.

[4] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[5] P. Sommer and R. Wattenhofer. Symmetric Clock Synchronization in Sensor Networks. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2008.

[6] R. Fan and N. Lynch. Gradient Clock Synchronization. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, 2004.

[7] T. Locher and R. Wattenhofer. Oblivious Gradient Clock Synchronization. In *20th International Symposium on Distributed Computing (DISC)*, 2006.

[8] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[9] J. Lundelius and N. A. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2/3):190–204, 1984.

[10] T. K. Srikanth and S. Toueg. Optimal Clock Synchronization. *J. ACM*, 34(3), 1987.

[11] R. Ostrovsky and B. Patt-Shamir. Optimal and Efficient Clock Synchronization Under drifting Clocks. In *PODC '99: Proceedings of thee ighteenth annual ACM symposium on Principles of distributed computing*, 1999.

[12] L. Schenato and G. Gamba. A distributed consensus protocol for clock synchronization in wireless sensor network. *46th IEEE Conference on Decision and Control*, 2007.

[13] D. Mills. Internet Time Synchronization: the Network Time Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, Oct 1991.

[14] K. Romer. Time Synchronization in Ad Hoc Networks. In *Mobi Hoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001.

[15] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *OSDI '02: Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.

[16] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync Protocol for Sensor Networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

[17] J. Sallai, B. Kusy, A. Ledeczi, and P. Dutta. On the scalability of routing integrated time synchronization. *3rd European Workshop on Wireless Sensor Networks (EWSN)*, 2006.

[18] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[19] L. Meier and L. Thiele. Brief announcement: Gradient clock synchronization in sensor networks. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, 2005.

[20] C. Lenzen, T. Locher, and R. Wattenhofer. Clock Synchronization with Bounded Global and Local Skew. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[21] J. Wolfowitz. Products of Indecomposable, Aperiodic, Stochastic Matrices. *Proceedings of the American Mathematical Society*, 14(5), 1963.

[22] M. Cao, A. S. Morse, and B. D. O. Anderson. Reaching a Consensus in a Dynamically Changing Environment: Convergence Rates, Measurement Delays, and Asynchronous Events. *SIAM J. Control Optim.*, 47(2), 2008.

[23] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Senso0072 Networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

.

