Hardware Validated Efficient and Simple Time Synchronization Protocol for clustered WSN

G S S Chalapathi[1], Raunak Manekar[1], Vinay Chamola[2], K.R.Anupama[3], S Gurunarayanan[1]
[1]Department of EEE/E&I, BITS-Pilani, Pilani, Rajasthan, India
[2]Department of Electrical and Computer Engineering National University of Singapore, Singapore
[3]Department of EEE/E&I, BITS-Pilani K K Birla Goa Campus, Goa, India

***Abstract*—Time Synchronization in WSN is a well-researched topic. But there are very few algorithms which have been proven to be scalable and efficient on a hardware platform. We consider cluster-based network and try to improve the efficiency of the existing algorithms. Our algorithm called as Simple Algorithm for Time Synchronization (SATS) is based on an existing receiver-sender synchronization algorithm [1]. We implement it in a cluster-based topology and do its performance analysis on a large WSN testbed. Its performance is compared with commonly used regression based synchronization protocol both analytically and on the hardware platform. Our experiments show that our algorithm shows a significant improvement in terms of average synchronization error and computational efficiency over regression based method.**

Keywords—Time synchronization, Wireless Sensor Networks (WSNs), cluster-based WSN

I. INTRODUCTION

Advances made in semiconductor technology especially in last 25 years have paved way not just for an increase in computational power of the existing processor technology but has also led to their miniaturization. This has powered the extensive research carried out in Wireless Sensor Networks (WSN) over the last two decades. WSNs are poised to play a significant role in the upcoming Internet of Things (IoT). WSN being a distributed network needs a unified view of time either at network level or at least for a group of nodes as per the application requirements. Thus time synchronization protocols are essential in a generic WSN and they play an important role in many network functionalities like data fusion, localization algorithms, time based duty cycling, channel access scheduling, etc. Given the fact that WSN nodes are cheap, battery powered and deployed in harsh environments, the time synchronization protocols developed must be energy-efficient, tunable as per the application requirements. Also factors like cheap crystal oscillators used in the WSN nodes, frequency variation of these crystals due to temperature and other environmental conditions, crystal aging, effect of sleep-wake cycles on the timer functionalities, etc. demand a need for periodic synchronization of nodes.

Time synchronization in WSN is a well-researched topic with numerous algorithms published by the scientific community over the last 15 years. Some of these algorithms like TPSN [2], RBS [3], FTSP [4], etc. report a microsecond-level accuracy in synchronization; while others like SLTP [5], L-SYNC [6] show a millisecond accuracy. However, one of the major factors in devising a time-synchronization protocol for a large scale network is its scalability which depends on its efficiency. To the best of our knowledge there are very few algorithms which have been tested and proven to be scalable on a large WSN. Works like SLTP [2], L-SYNC [6] have shown the potential to be scalable. But they are simulation based works and their performance on a hardware platform is not proven. So we set out with a goal to devise a potentially scalable Time-synchronization protocol and test it on a large scale hardware testbed. Also we would limit our scope of our work to a cluster based topology. Cluster based topology is one of the most efficient topologies in WSN offering energy efficiency while gathering data, route formations, etc. [6] [7].

The rest of the paper is organized as follows: firstly, we would discuss the state of the art relevant to our work. We next describe the algorithm that we will be using in our work in section III. The experimental setup, the methodology and results would be discussed in the next section. We finally offer our conclusions and discuss about the future work in the section V.

II. RELATED WORK

Since we are dealing with the time-synchronization problem, it is essential to define the clock model that we are considering in this work. We consider a clock-skew model as given by the following equation

$$C_n(t) = \alpha_n . t + \beta_n \qquad (1)$$

where t is the absolute time in the network, α_n is the absolute skew and β_n is the offset this n^{th} node with respect to an absolute clock. If we want to calculate the local time of the cluster member with respect to the cluster head as we would do in our work, then (1) would transform into the following

$$C_i(t) = \alpha_i . C_h + \beta_i \qquad (2)$$

where α_i is the relative skew, β_i is the relative offset of the i^{th} node with respect to its cluster head and C_h is the local time of the cluster head.

As mentioned before we limit our scope to time-synchronization in clustered WSN. Some of the works which address time synchronization in clustered WSN are SLTP [2], L-SYNC [6] and PC_Avg [8]. In PC_Avg the time stamps of the cluster members are collected by the cluster head which in turn calculates the average of the timestamps and broadcasts it to its

978-1-5090-2597-8/16/$31.00 ©2016 IEEE

cluster members. SLTP [5] and L-SYNC [6] on the other hand uses regression on the local time stamps received from the cluster members. Both of these algorithms differ in the way they form clusters.

PC_Avg suffers from very high synchronization error of the order of tens of seconds. Further it does not take into account the non-deterministic and deterministic delays that occur during the transmission of packets in a WSN. Also SLTP [5] and L-SYNC [6] do not account for these delays and thus also suffer from very high synchronization errors.

There are different sources of delays like send time, access time, transmission time, propagation time, reception time, receive time, etc. [2] [3] [4]. These delays significantly affect the synchronization protocol that we use as the magnitude of these delays can be of the order of milli-seconds [4]. We can broadly classify these delays as deterministic and non-deterministic delays as done in Chaudari et al. [1]. We build our work based on this work by adapting it to a cluster-based WSN and demonstrate its performance on a hardware platform. We further on refer this proposed algorithm used in a clustered WSN as Simple Algorithm for Time Synchronization (SATS).

Many clock synchronization algorithms such as [9], calculate only the relative offset between clocks of the two nodes assuming no clock skew. This leads to clocks going out of synchronization very soon and thus needing very frequent synchronization events. In SATS and the algorithms that we compare, we estimate both the clock skew and offset.

SATS like in [1] considers a two-way exchange between two nodes in a network as shown in Fig. 1. Two nodes which are being synchronized with each other exchange two packets which are time stamped when sent and received. These time stamps are used to determine the relative skew and offset of a node with respect to the other. The time stamps T_1, T_2, T_3 and T_4 are related to each other by the set of following equations.

$$T_{2,k} = \alpha\,(T_{1,k} + t_f + r_k) + \beta \qquad (3)$$

$$T_{3,k} = \alpha\,(T_{4,k} - t_f - s_k) + \beta \qquad (4)$$

Where, $T_{1,k}$, $T_{2,k}$, $T_{3,k}$ and $T_{4,k}$ are the time stamps of the k^{th} packet; t_f denotes the fixed delay incurred in the packet exchange in absence of any random delays. r_k and s_k are the nondeterministic delays which occur in the packet exchange process and introduce unpredictability in the system. SATS does not require us to model these random delays as Gaussian or exponential distributed random variables. This is required for many other estimator methods such as maximum likelihood described in [1] and [10].

III. ALGORITHM

Since both SLTP and L-SYNC which are the prominent time synchronization algorithms and have shown to be scalable (on simulator) are based on regression, we compare SATS with a generic regression based time synchronization method. For comparison we have used both linear regression and SATS to calculate the synchronization error between the sender and the receiver nodes. Linear regression has been traditionally used successfully in many estimation problems like [3], [4], [5]. But its high computational complexity makes it unsuitable for embedded applications. Many applications require time stamps with microsecond accuracy. With this increased clock resolution, the size of the timestamp variable needs to be increased to measure a considerable time before it overflows. Obtaining microsecond timestamps of 32-bit width allows measurement of a duration up to 1 hour. The large number of 32-bit multiplications and additions to be performed are very difficult for common WSN processors like MSP430 of TelosB motes. More powerful motes like Lotus or Wasp, with 32-bit processors can perform such high computations but with very high power consumption. It is not preferable to allocate so much of power and processor time for time synchronization protocols. Also, very high cost of these powerful motes makes it unfeasible to use them in large numbers in a WSN.

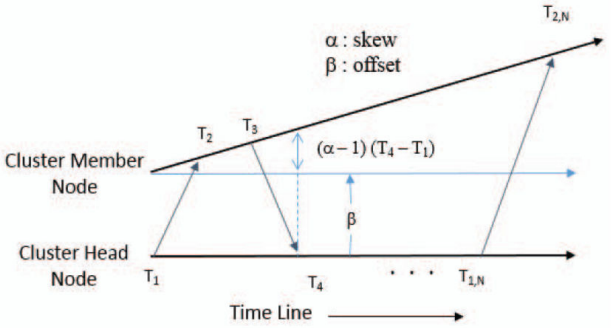

Fig.1. Two- way exchange between a two nodes with the timestamps

SATS drastically reduces the computational complexity as compared to regression. We compare SATS and regression in Table I. We see that SATS is efficient both in terms of additions and multiplications performed during synchronization. Moreover, the multiplications performed to get a microsecond accuracy are 32 bit multiplications which require lot of computation power thus giving a significant upper hand to SATS over traditional regression based method. Also this makes SATS suitable for deployment in low-power motes and very useful for creating a scalable time synchronization protocol. The simulations performed in [1], shows that the algorithm gives acceptable results in terms of MSE even when compared to highly complex Maximum-likelihood estimate based methods in two nodes. We tested both linear regression and SATS on data collected from physical motes. In the considered cluster-based topology, we are synchronizing the cluster members with the cluster head. The computations for both SATS and regression are performed on a PC after collection of the timestamps from the hardware testbed.

As shown in figure, for a sent packet, T_1 and T_2 represent the send time on the sender node and receive time on the receiver node respectively. The two timestamping events are separated by the propagation delay (d) and the non-deterministic delays involved in transmitting and receiving the synchronization packets. This gives rise to a significant error in between the actual local time on receiver node and the time of the sender calculated using the estimated clock skew and offset. SATS tries to identify the two most accurate timestamp samples among those obtained from the two-way message exchanges performed N times. It then uses them to calculate the skew and offset

between the clocks, as discussed below. Accuracy of timestamps referred here is with respect to time delay between the two time-stamping events.

TABLE I. ALGORITHM COMPLEXITY

Algorithm	*Additions*	*Multiplications*
Regression	4N + 3	2N + 6
SATS	3N + 7	1

By rearranging the equations (3) and (4) of the timing message exchange model, we can see that the non-deterministic delays tend to increase the overall duration of two-way message exchange. Therefore, the smaller the duration of a two-way packet exchange (calculated by T_4-T_1), the lesser random delays it has suffered and more accurate the timestamps.

$$T_{2,k} = \alpha T_{1,k} + \beta + \alpha (t_f + r_k)$$

$$T_{3,k} = \alpha T_{4,k} + \beta - \alpha (t_f + s_k)$$

In SATS, two samples (of the N exchanges) with most accurate time-stamps are identified by selecting the ones with minimum (T_4- T_1) duration. And the timestamps from these two samples are used to calculate the skew and offset between the two clocks [1]. Thus, the calculation is affected only by the recorded timestamps with the least error. On the other hand, regression tries to obtain a best fit line from the data and is thus affected by all the points regardless of the error in the timestamp values. This is an intuitive explanation for the high synchronization accuracy achieved by SATS at very low computation costs.

So each cluster member performs N number of two-way exchanges with the cluster head and thus calculates its relative skew and offset with respect to the cluster head and gets synchronized with it. Thus at the end of a synchronization cycle all the nodes in the cluster are synchronized to the cluster head.

IV. EXPERIMENTAL PERFORMANCE

A. Experimental Setup

All experiments have been performed on MSP430 based TelosB [11] motes using TinyOS 2.1.2 [12]. All the nodes in the network have been spread out with a density of 5 nodes per 5 square meters' area. This density is very high as compared to any real life WSN network. Testing both the algorithms in this extreme scenario gives one of the worst case performance estimates. The experiment was conducted in an environment with competing technologies such as Wi-Fi also operating at the same 2.4 GHz frequency so as to ensure some wireless traffic within the channel of communication. A photograph of part of the network deployed is shown in Fig. 2.

B. Methodology

A two-way packet exchange is performed between two motes. The sender mote (say A) sends a synchronization request to the receiver mote (say B) and records the send-timestamp of the packet. Then node B records the time at which the packet was received and sends an acknowledgement with receive and resend timestamps. Then node A records the receive time of the

Fig. 2. A photograph of a part of the network deployed

acknowledgement packet. One such exchange gives us a set of 4 timestamps (T_1, T_2, T_3, and T_4). These timestamps are used by the sender node (i.e. cluster member) to calculate the skew and offset of the receiver node (i.e. cluster head) clock with respect to its own clock. In the following discussion, we will refer to this relative skew and relative offset as skew and offset respectively. This experiment is repeated in a cluster with the cluster head trying to synchronize with 5 cluster member nodes. Further, experiments were performed with a number of other clusters (5 motes each) in the vicinity which are themselves running the time synchronization protocol thus contributing to significant overall traffic in the network. In each case, the synchronization error was calculated using both the methods (regression and SATS) to compare their performance. The timestamps were collected from the WSN motes and the evaluations were performed on the PC for both SATS and regression.

Further, to evaluate the synchronization accuracy achieved, after the nodes are synchronized, we check the synchronization error for timestamps of a common event. There are two ways of creating this common event. One way is to send an $(N+1)^{th}$ packet from the cluster head (say) after N training packets and try to estimate on the cluster member its T_1 value i.e. time of sending the packet by the cluster member receiver based on its T_2 value (time of receiving the packet at the cluster member). But again this method would suffer from errors due to delays caused in the transmission of this $(N+1)^{th}$ packet and thus cause errors in synchronization error estimation. Another method which we have used in this work is to use a third party node (which is not part of the cluster of nodes to be synchronized), to broadcast a packet to all the nodes in the cluster including the cluster head. It is reasonable to assume that this broadcast is received by all the nodes simultaneously as the non – deterministic delay in packet transmission is almost the same for all nodes since nodes of same cluster will be close by. And MAC-layer timestamping further reduces the random delays on receiver side like in pervious works e.g. TPSN [3]. Thus the reception of this broadcast gives us a common event whose timestamps can be collected at the cluster head and used to evaluate the synchronization error.

C. Result Analysis

Microsecond accuracy of time synchronization was observed in all experiments with both regression and SATS.

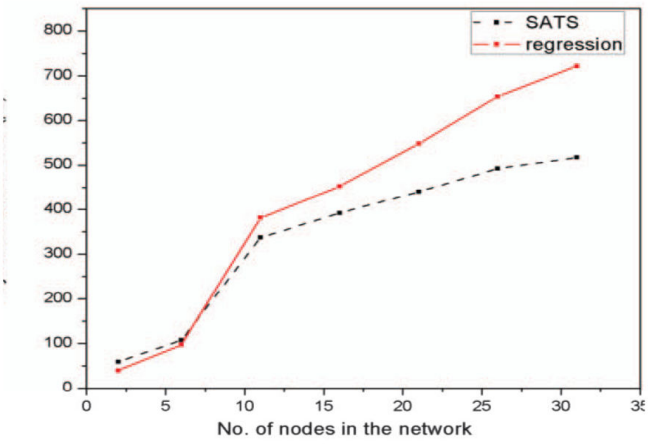

Fig. 3. Average synchronization error vs number of nodes in the network for SATS and regression based synchronization

From Fig. 3, we see that with increase in the size of the network, synchronization error increased from around 40 µs to nearly 500 µs for SATS and to 700 µs for regression based method. The reason for this is as the size of network increases, large traffic leads to increased packet collisions. Thus the resultant loss of synchronization packets reduces the accuracy of time synchronization achieved by both the methods. The large error values for larger networks are result of the very high network density chosen for the experiment.

For small sized networks, regression gives less synchronization error as compared to SATS. But as the size of the network increases, SATS outperforms regression. This can be observed from Fig. 4. This is because SATS is less affected by the loss of packets because it considers only the two packet exchanges with least durations. There is a steady increase in the percentage improvement made by SATS over regression based method with network size due to the above mentioned reason.

It is also observed that synchronization error varies widely across multiple experiments on the same network of fixed size. Therefore, average synchronization error and percentage improvement therein does not give a complete picture of the comparison of the performance of the two methods. Hence, we calculate the percentage of experiment trials SATS gives better or comparable accuracy as compared to regression.

$$p_n = \frac{no.of\ synchronization\ events\ where\ SATS\ gives\ better\ or\ comparable\ result}{total\ no.\ of\ synchronization\ events} \times 100$$

n = size of the network (no. of nodes). n is fixed over the considered set of experiments.

Cases where SATS is giving comparable results are also considered in calculating this percentage because same synchronization error is obtained at a drastically reduced computation cost. This feature makes the SATS preferable. Comparable result has been defined as a case where the difference of absolute values of synchronization error obtained using the two methods is within a margin of 10% of the average synchronization error for that particular size of network. E.g. the average synchronization error obtained in a network of size 16 is approximately 450 µs. A case where synchronization error given by SATS within a margin of 45 µs is counted as comparable.

This measure gives a better evaluation of performance of the two algorithms. And as we can see in Fig. 5, that SATS performs better than regression in most of the cases for large networks where this percentage touches 85%. Also significant to note is that even in small sized networks when average synchronization error of regression based method is lesser than SATS, about 60-70% of trial were comparable (as in 6 node network).

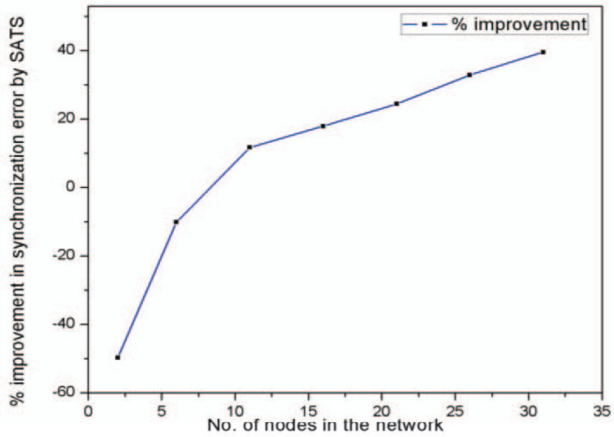

Fig. 4. Percentage improvement in synchronization error made by SATS over regression based method with increase in network size

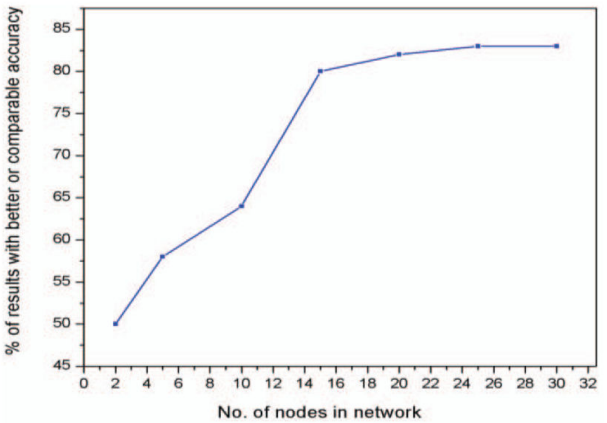

Fig. 5. Percentage of results in which SATS shows a better or comparable synchronization accuracy compared to regression based method with increase in network size

V. CONCLUSION AND FUTURE WORK

Thus we demonstrated on a hardware platform a simple but efficient synchronization method which outperforms traditional regression based method. This improvement becomes more

prominent in large dense network with large traffic. Though SATS uses more number of packets compared to regression method during synchronization, the reduction in the synchronization error would reduce the number of synchronization cycles required in SATS and thus prove to be efficient in the long run. We want to demonstrate this in our future work. Also we expect that when computations are performed on the nodes rather than on PC, regression method would take longer time to synchronize the network. We also want to demonstrate this in our future work.

ACKNOWLEDGMENT

We acknowledge Mr. Akshay Madan for his inputs during our discussions of this algorithm.

REFERENCES

[1] Chaudhari, Qasim M., Erchin Serpedin, and Khalid Qaraqe. "On maximum likelihood estimation of clock offset and skew in networks with exponential delays." IEEE Transactions on Signal Processing 56.4 (2008): 1685-1697.

[2] Ganeriwal, Saurabh, Ram Kumar, and Mani B. Srivastava. "Timing-sync protocol for sensor networks." Proceedings of the 1st international conference on Embedded networked sensor systems. ACM, 2003.

[3] Elson, Jeremy, Lewis Girod, and Deborah Estrin. "Fine-grained network time synchronization using reference broadcasts." ACM SIGOPS Operating Systems Review 36.SI (2002): 147-163.

[4] Maróti, Miklós, et al. "The flooding time synchronization protocol." Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, 2004.

[5] Gelyan, Sepideh Nazemi, et al. "SLTP: scalable lightweight time synchronization protocol for wireless sensor network." International Conference on Mobile Ad-Hoc and Sensor Networks. Springer Berlin Heidelberg, 2007.

[6] Jabbarifar, Masoume, et al. "L-SYNC: larger degree clustering based time-synchronisation for wireless sensor network." Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on. IEEE, 2010.

[7] Heinzelman, Wendi Rabiner, Anantha Chandrakasan, and Hari Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks." System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on. IEEE, 2000.

[8] Mamun-Or-Rashid, Md, Choong Seon Hong, and Chi-Hyung In. "Passive cluster based clock synchronization in sensor network." Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05). IEEE, 2005.

[9] Jeske, Daniel R. "On maximum-likelihood estimation of clock offset." IEEE Transactions on Communications 53.1 (2005): 53-54.

[10] Wu, Yik-Chung, Qasim Chaudhari, and Erchin Serpedin. "Clock synchronization of wireless sensor networks." IEEE Signal Processing Magazine 28.1 (2011): 124-138.

[11] Memsic Inc. Telosb mote platform. Datasheet. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

[12] TinyOS Homepage. [Online]. Available: http://www.tinyos.net/