The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

Improved SNTP for Accurate Time Synchronization in Smart AMR systems

Se-Young Oh, Dong-Doo Lee, Chang-Hwa Lee

ADMOTECH Inc, 1320 Gwanpyung-Dong Yusong-Gu, Dagjon 305-509, South Korea
(Tel : 82-42-936-1353; Fax : 82-42-936-1350)
E-mail : {syoh, ddlee, chlee}@admotech.com

Abstract : In distributed sensor network such as distributed AMR system, accurate time synchronization is necessary to assure the concurrence of event timing for measured data. There are NTP(Network Time Protocol) and SNTP (Simple Network Time Protocol), RBS (Reference Broadcast Synchronization), TPSN (Time synchronization Protocol for Sensor Networks) in time synchronization for distributed network system**s.**

In this paper, we suggested improved SNTP using precise meta data exchange and agile interrupt handling techniques and showed that our method has accuracy of sub-millisecond compared to conventional SNTP with accuracy of few second from the time synchronization performance analysis results.

Keywords : Time Synchronization, SNTP

I. INTRODUCTION

As in distributed sensor network such as distributed AMR systems, accurate time synchronization is necessary to assure the concurrence of event timing for measured data.

There are two main purposes of time synchronization.

The first purpose is to ensure that events occur on time, in the correct sequence. Therefore, synchronization is necessary to start scheduled events and to register their occurrence. Many activities in commerce, banking, financial, business, transport, medicine, services, as a few examples, may need to guarantee that tasks are timely scheduled, and concurrent and cooperating processes interoperate correctly. The second purpose is tracing, that is, retrieving information concerning past events, whenever is necessary, regarding when the events occurred and in what sequence. This task is possible only if accurate timestamps of each event are available.

In this paper, we suggested improved SNTP using precise meta data exchange and agile interrupt handling techniques and showed that our method has accuracy of sub- millisecond compared to conventional SNTP with accuracy of few second from the time synchronization performance analysis results.

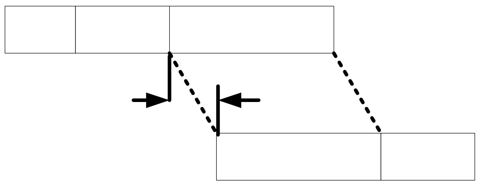
II. OVERVIEW[2]

All network time synchronization methods rely on some sort of message exchange between nodes.

Non-determinism in the network dynamics makes the synchronization task challenging in many systems.

When a node in the network generates a timestamp to send to another node for synchronization, the packet carrying the timestamp will face a variable amount of delay until it reaches and is decoded at its intended receiver. This delay prevents the receiver from exactly comparing the local clocks of the two nodes and accurately synchronizing to the sender node. We can basically decompose the sources of error in network time synchronization methods into basic components:

- ☐ Send Time: This is the time spent to construct a message at the sender. It includes the overhead of operating system, and the time to transfer the message to the network interface for transmission.
- ☐ Access Time: Each packet faces some delay at the MAC layer before actual transmission. The sources of this delay depend on the MAC scheme used, but some typical reasons for delay are waiting for the channel to be idle or waiting for the TDMA slot for transmission.
- ☐ Propagation Time: This is the time spent in propagation of the message between the network interfaces of the sender and the receiver.
- ☐ Receive Time: This is the time needed for the network interface of the receiver to receive the message and transfer it to the host.



[Fig. 1]. Message delivery delay

a. The Need For Synchronization in Sensor Networks

There are several reasons for addressing the synchronization problem in sensor networks.

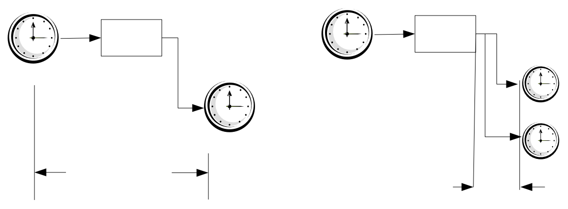
First, sensor nodes need to coordinate their operations and collaborate to achieve a complex sensing task. Data fusion is an example of such coordination in which data collected at different nodes are aggregated into a meaningful result. For example, in a vehicle tracking application, sensor nodes report the location and time that they sense the vehicle to a sink node which in turn combines these information to estimate the location and velocity of the vehicle. Clearly, if the sensor nodes lack a common timescale (i.e., they are not synchronized) the estimate will be inaccurate.

Second, synchronization can be used by power saving schemes to increase network lifetime. For example, sensors may sleep at appropriate times, and wake up when necessary. When using power-saving modes, the nodes should sleep and wake-up at coordinated times, such that the radio receiver of a node is not turned off when there is some data directed to it. This requires a precise timing between sensor nodes.

b. Reference Broadcast Synchronization (RBS)

Reference Broadcast Synchronization (RBS) [3], where their simple yet novel idea is to use a "third party" for synchronization, their scheme synchronizes a set of receivers with one another.

In RBS scheme, nodes send reference beacons to their neighbors. A reference beacon does not include a timestamp, but instead, its time of arrival is used by receiving nodes as a reference point for comparing clocks.



[Fig. 2] Comparison of conventional method to RBS

The authors argue that, by removing the sender's non-determinism from the critical path [Fig. 2], RBS achieves much better precision compared to traditional synchronization methods that use two-way message exchanges between synchronizing nodes. As the sender's non-determinism has no effect on RBS precision, the only sources of error can be the non-determinism in propagation time and receive time. The authors claim that a single broadcast will propagate to all receivers at essentially the same time, and hence the propagation error is negligible. This is especially true when the radio ranges are relatively small, as is the case for sensor networks. So they only account for the receive time errors when analyzing accuracy of their model.

In the simplest form of RBS, a node broadcasts a single pulse to two receivers. The receivers, upon receiving the pulse, exchange their receiving times of the pulse, and try to estimate their relative phase offsets. This basic RBS scheme can be extended in two ways:

1) allowing synchronization between n receivers by a single pulse, where *n* may be larger than two,

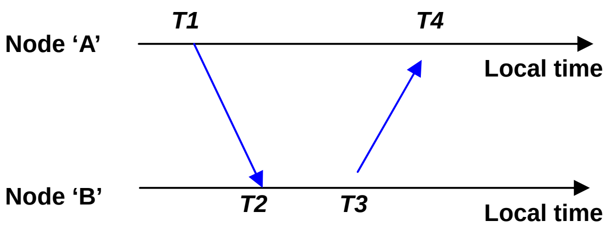
2) increasing the number of reference pulses to achieve higher precision. The authors show by simulation that 30 reference broadcasts (for a single synchronization in time) can improve the precision from 11 μs to 1.6 μs

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

Timing-Sync Protocol for Sensor Networks (TPSN) [4] works in two phases: level discovery phase" and synchronization phase". The aim of the first phase is to create a hierarchical topology in the network, where each node is assigned a level. Only one node is assigned level 0, called the root node. In the second phase, a node of level i synchronizes to a node of level $i-1$. At the end of the synchronization phase, all nodes are synchronized to the root node and the network-wide synchronization is achieved.

Consider a two-way message exchange between nodes A and B as shown in [Fig.3]. Node A initiates the synchronization by sending a synchronization pulse packet at $T1$(according to its local clock). This packet includes A's level number, and the value $T1$.

B receives this packet (according to its local clock) at $T2 = T1 + \Delta + d$, where Δ is the relative clock drift between the nodes, and d is the propagation delay of the pulse. B responds at time $T3$ with an acknowledgement packet, which includes the level number of B and the values $T1$, $T2$, and $T3$. Then, node A can calculate the clock drift and propagation delay as below, and synchronize itself to B.



[Fig. 3] Two way message exchange between a pair of nodes

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2}; \qquad \text{Eq. 1.}$$

$$d = \frac{(T2 - T1) + (T4 - T3)}{2}; \qquad \text{Eq. 2.}$$

The synchronization phase is initiated by the root node's time sync packet. On receiving this packet, level 1 nodes initiate a two-way message exchange with the root. Before initiating the message exchange, each node waits for some random time, in order to minimize collisions on the wireless channel. Once they get back a reply from the root node, they adjust their clocks to the root node. Level 2 nodes, overhearing some level 1 node's communication with the root, initiate a two-way message exchange with a level 1 node, again after waiting for some random time to ensure that level 1 nodes have completed their synchronization. This procedure eventually gets all nodes synchronized to the root node. TPSN is implemented on Berkeley's Mica architecture, and makes use of time-stamping packets at the MAC layer in order to reduce uncertainty at sender. Ganeriwal et.al. claim that TPSN achieves two times better precision than RBS. They state that the precision of 6.5 μs .

d. Requirements on the Synchronization Schemes for Sensor Networks

There are trade off between the requirements of an efficient synchronization solution (e.g., precision versus energy efficiency), thus a single scheme may not satisfy them altogether.

☐ Energy Efficiency : As with all of the protocols designed for sensor networks, synchronization schemes should take into account the limited energy resources contained in sensor nodes.

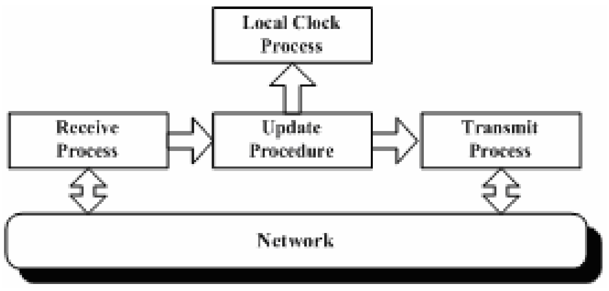
☐ Scalability - Most sensor network applications need deployment of a large number of sensor nodes. A synchronization scheme should scale well with increasing number of nodes and/or high density in the network.

☐ Precision - The need for precision, or accuracy, may vary significantly depending on the specific application and the purpose of synchronization. For some applications, even a simple ordering of events and messages may suffice whereas for some others, the requirement for synchronization accuracy may be on the order of a few μs .

☐ Cost and Size - Wireless sensor nodes are very small and inexpensive devices. Therefore, as noted earlier, attaching a relatively large or expensive hardware on a small, cheap device is not a logical option for synchronizing sensor nodes. The synchronization method for sensor networks should be developed with limited cost and size issues in mind.

III. CONFICURATION

In Smart AMR system we would like to assure the concurrence of event timing for measured data within milliseconds. Concerning the trade-off no hardware and bandwidth added to basic PHY.



[Fig 4] Implementation Model

Followings are major component consist of our system configuration.

- Two-way time transfer algorithm
- One millisecond counting RTC
- Agile interrupt handler
- Simple moving average digital filter

The moving average is the most common digital filter to understand and use. In spite of its simplicity, the moving average filter is optimal for a common task: reducing random noise while retaining a sharp step response. This makes it the premier filter for time domain encoded signals.

As the name implies, the moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. In equation form, this is written:

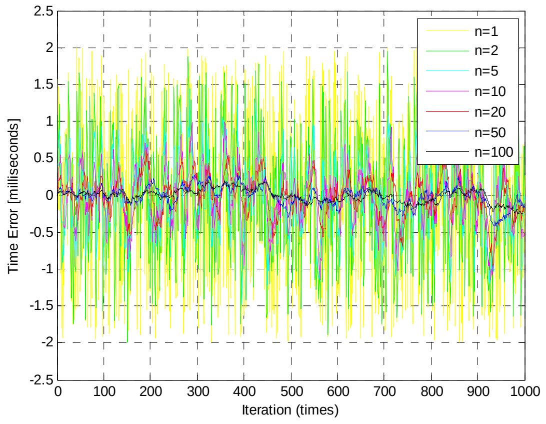
$$y[i] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j] \qquad \text{Eq. 3.}$$

Where $x[]$ is the input signal, $y[]$ is the output signal, and M is the number of points in the average.

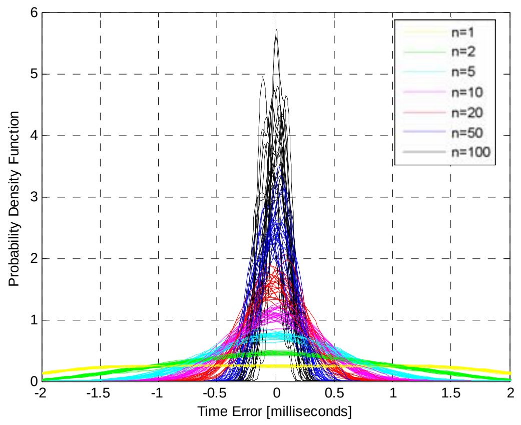
IV. ANALYSIS

[Fig. 5] shows the effect of how this works reduce timing fluctuation. The signal colored yellow is timing raw data. The other color's plots show moving averaged result, the smoothing action of the moving average filter decreases the amount of the random time variance. The amount of noise reduction is equal to the square-root of the number of points in the average. For example, a 100 point moving average filter reduces the noise by a factor of 10.

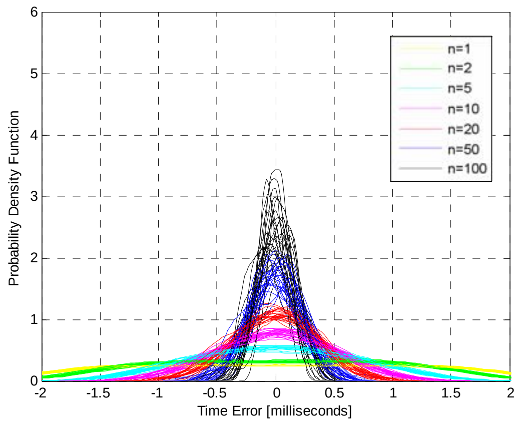
[Fig. 6] and [Fig. 7] show probability density function for simple moving average and square weighted moving average respectively.



[Fig. 5] Time Synchronization error smoothing characteristics for variable average numbers.



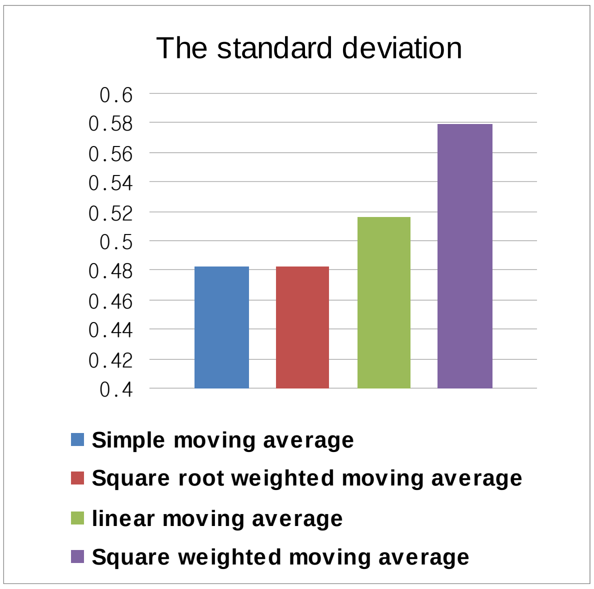
[Fig. 6] Probability density function for simple moving average



[Fig. 7] Probability density function for square weighted moving average

V. CONCLUSION

There are trade off between the requirements (eg. Energy Efficiency, Precision, Cost and Size). We constructed improved SNTP using precise meta data exchange and agile interrupt handling techniques with simple moving averaging algorithm. It shows 0.48 millisecond jittering performance is obtained with 100 points in the average.



[Fig. 8] Jittering Characteristics
Standard deviation vs. averaging algorithm

ACKNOWLEDGEMENTS

The Work developed in this paper has been supported by the DAEDEOK INNOPLIS ("R&D Hub Cluster project").

REFERENCES

[1] Cristina D. Murta, Pedro R. Torres-Jr., "Characterizing Quality of Time and Topology in a Time Synchronization Network"

[2] Fikret Sivrikaya et. al., "Time Synchronization in Sensor Networks: A Survey" IEEE In Network, IEEE, Vol. 18, No. 4. (2004), pp. 45-50.

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

[4] S. Ganeriwal, Ram Kumar, M. Srivastava, \Timing Sync Protocol for Sensor Networks", ACM SenSys, Los Angeles, November 2003.

[5] D. Mills, Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, University of Delaware, 1996

[6] Steven W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing" Second Edition, California Technical Publishing, San Diego, 1999, *pp 277-279*