Multi-Hop Synchronization at the Application Layer of Wireless and Satellite Networks

A. Marco[a,1], R. Casas[a], J. L. Sevillano[b], V. Coarasa[a], J. L. Falcó[a] and M. S. Obaidat[c], Fellow, IEEE
[a] Tecnodiscap group at the University of Zaragoza, Spain ([1]) amarco@unizar.es.
[b] Dept. of Computer Architecture and Technology. University of Seville. Spain.
[c] Computer Science Dept. Monmouth University, NJ, USA. E-mail: obaidat@monmouth.edu

***Abstract*—Time synchronization is a key issue in wireless and satellite networks; time-stamping collected data, tasks scheduling or efficient communications are just some applications. From all the existing techniques to achieve synchronization, those that work at the MAC layer and can precisely timestamp sync messages are the most accurate. However, working with standard protocols, usually prevents the user from accessing lower layers and consequently reduces accuracy. Receiver—receiver schema improves time-stamping performance because it eliminates the biggest non-deterministic error at the sender side; the medium access time. Nevertheless, utilization of these methods in multi-hop networks usually requires an extra amount of traffic. In this paper we present a method which allows accurate synchronization of large multi-hop networks such as satellite networks working at the application layer while keeping the message exchange to the minimum. Through an exhaustive experimentation, we show the protocol's performance and analyze the factors that influence synchronization accuracy the most.**

I. Introduction

Time synchronization entails an important function in wireless and satellite networks, and this function can be performed at different layers. For instance, sharp timing is fundamental at low layers to increase data rates (short bit times), enhance noise immunity (frequency hopping) and TDMA (Time Division Multiple Access) scheduling. Furthermore, since radio is usually the most energy-consuming part in a node, keeping the nodes awake the minimum required time in order to exchange data is a common practice that requires synchronization [1].

Synchronization is also of great interest at higher layers. Akyildiz et al. specify the duties to be performed in each layer of protocol stacks used in sensor networks. They identify in the application layer the *sensor management* protocol, which includes time synchronization [2]. One of the most important functions of a sensor network is *data fusion* [3], which needs synchronization for two tasks: time scheduling and time stamping. The first is needed when the nodes coordinate to perform cooperative communications. The second is commonly used when data is fused taking into account the collecting instant; for example, to perform event detection, tracking, reconstruction of system's state for control algorithms, off line analysis, etc. On the other hand, in satellite networks clock synchronization is also very important for many applications, e.g. accurate phase evaluation in interferometric synthetic aperture radar (InSAR) systems [4].

GPS is a classical solution for synchronization, but it has some drawbacks. In sensor networks, the cost of the dedicated hardware needed, the settling time (that could take up to several minutes) and the need of a clear sky view are some of them. In satellite networks, there are also situations where clock synchronization using GPS maybe impossible or insufficient, for instance when working at Low Earth Orbit (LEO) and beyond [4].

Creating a common temporal reference using the nodes communication capabilities has been widely studied; keeping in mind the energy, cost and size limitations of the devices used in sensor networks [3, 5]. Synchronization methods can be classified based on many different criteria [5,6]. For instance, with *a posteriori* synchronization methods devices' clocks run free, gathering information between relative clocks and rearranging timestamps once the measurement processes are finished. These methods are usually the most energy-efficient because they optimize the number of messages exchanged, but they don't offer real time capabilities. On the other hand, *a priori* methods overcome this synchronizing issue at all the nodes with a common time reference (global network time – GNT) by regular clock corrections. A common drawback of these techniques is the overload of the network due to the messages necessary to estimate the communication delays.

Another key issue is whether there is a sender transmitting the current clock values as timestamps (*sender-to-receiver*) or not (*receiver-to-receiver*). The problem with sender-to receiver methods is the uncertainty time introduced by the send and access processes. In receiver-to-receiver methods, the use of reference broadcast messages to establish a common time reference gets rid of the transmitter-side non-deterministic error sources (it is assumed that all devices listening to the broadcast get the message at the same time) [6]. The biggest drawback is how to propagate the local time-stamps of the broadcast-receivers to set a GNT.

Reference Broadcast Synchronization (RBS) [6] is a receiver-to-receiver method that performs synchronization only when it is needed through a post-facto synchronization [7]. Originally proposed for wireless sensor networks (WSNs), it has also been used in satellite networks [4]. The scattering method proposed by authors in [7] does not give a common

This work was supported in part by the Spanish MCYT under AmbienNET project (TIN2006-15617-C03-02) and in part by the European Union under projects MonAmI and EasyLine+.

time reference to the broadcast sender; it only synchronizes receivers. However, various authors point out to the need of setting a network time to propagate the synchronization over a multi-hop network using broadcast [6, 8]. Thus, nodes in broadcast domain need to share timing information among them in order to determine the GNT. In large networks, the amount of data exchange required is huge [9].

On the other hand, TPSN (Timing-sync Protocol for Sensor Networks) [10] is a sender-to-receiver protocol that avoids the indeterminism working at the MAC (Medium Access Control) layer to precisely timestamp messages at the exact moment they are sent. The Flooding Time Synchronization Protocol (FTSP) also uses MAC layer time-stamping at both the sender and the receiver sides. The protocol proposes a multi-hop propagation scheme that does not need any initial configuration to propagate synchronization info [11]. However, these sender—receiver synchronization schemes require accessing the lower layers, which is not always possible when using standard or complex protocols.

In this paper, we present a Multi-hop Broadcast Synchronization (MBS) method, a receiver—receiver synchronization scheme that nonetheless obtains a global network time working at the application layer. MBS is suitable for large multi-hop networks; keeping the number of messages in the same order of magnitude when compared to the sender—receiver methods. This helps to achieve high accuracy and energy-efficiency. It maybe useful when time-stamping at lower layers is not possible in multi-hop networks. In Section II, MBS is described. First, we introduce single-hop synchronization; discussing how clock offset and drift is treated to obtain a global clock using the minimum number of broadcast messages. Then we extrapolate the method to multi-hop networks. In Section III, we evaluate the method in two different standard platforms —Bluetooth and ZigBee— and compare it with other schemes used for both single and multi-hop synchronization. In this section, we also perform comprehensive experimentation to determine the factors that influence synchronization accuracy the most. Finally, section IV concludes the paper.

II. MULTI-HOP BROADCAST SYNCHRONIZATION PROTOCOL

Nodes in a WSN measure time with oscillators at slightly different rates. The variation can go up to 150 PPM, i.e. each second the nodes will commit an error that can go up to 150 μs. Apart from the clock skew, there is an offset among clocks because each node starts at a different instant. The clock reading of each node can be written as: $t_i = s_i t + k_i$, with $i = 1 \ldots n$. Thus, synchronizing the clock of a node i implies estimating and compensating clock skew and offset (s_i, k_i). The most used procedure to perform these adjustments is broadly described in the literature [3, 6, 11, 12]. There is a reference clock t_i to which all the nodes will be synchronized. A sync-point is defined as a pair of timestamps collected at the same time t^k in the reference node and in the node that wants to be synchronized: $\{t_i^k, t_r^k\}$. Once each node stores several sync-points at different instants, the offset and slope (s_i^*, k_i^*) differences with the reference can be calculated using linear regression:

$$s_i^* = \frac{\sum_k \left(t_r^k - \overline{t_r}\right)\left(t_i^k - \overline{t_l}\right)}{\sum_k \left(t_r^k - \overline{t_r}\right)^2};\ k_i^* = \overline{t_l} - s_i^* \overline{t_r} \qquad (1)$$

This way, every node can estimate the global time t_r^* from its local clock $t_r^* = (t_i - k_i^*)/s_i^*$. Once a node is synchronized, it can propagate the estimated t_r^* to others creating new sync-points that spread the GNT in multi-hop networks [11].

Strictly speaking, it is not possible to obtain a sync-point at the same instant in two nodes not physically linked; there will be unknown delays in the synchronization message exchange. While deterministic uncertainties in wireless links slightly affect the synchronization accuracy, nondeterministic ones drastically reduce it [2, 6, 10, 11]. TPSN and FTSP eliminate the biggest uncertainties (send, receive and access time) by time-stamping at the MAC layer the send and receive instant of a message. This way, to obtain the required sync-points accurately, only one message is necessary.

When access to the low layers is not possible, reference broadcast messages eliminate the uncertainty in the sender-side. Unfortunately, although reception instant of a broadcast message is tight, this procedure does not provide the required sync-points directly because the sender is not synchronized [6]. To our best knowledge, existing methods require additional message exchange between every receiver node to set the GNT; making multi-hop propagation very inefficient [6, 8].

Our MBS protocol obtains sync-points using reference broadcast with considerably less message overhead than other methods that also use reference broadcast; as also manages to synchronize all the nodes (even the broadcast sender) making multi-hop propagation easy.

A. Protocol Description

Nodes in MBS protocol perform two different tasks. *Propagators* are those that spread the GNT broadcasting synchronization messages. *Time-stampers* are nodes that can notify the propagators about the timestamp when the previous broadcast message arrives. In Fig. 1, N3, N6, N9 and N11 are propagators. N1, N7 and N10 are time-stampers; the provided timestamps are referred to the GNT, so they have to be synchronized when notifying the corresponding propagator node. To overcome this requirement when initiating the process, N1 will be the node whose local clock will be the GNT, i.e., N1 is the global time provider (GTP).

The Synchronization will be done using two different messages: *SyncBC* and *TimeUC*. First type is broadcast sent by propagators and is equivalent to the reference broadcast in RBS protocol: message that triggers time-stamping of the receiving instant at the sender-side. It contains three fields: the *propagatorID* identifying the sender, the *sequenceNumber* of the message and the *timeStamp* when the previous SyncBC arrived. These messages are used to propagate the GNT the same way as synchronization messages in FTSP [11]. TimeUC messages are unicast messages used by time-stampers to notify the propagators about the time when the last SyncBC

arrived. They have the following fields: the *timeStamperID* identifying the sender and the corresponding *propagatorID*, the *sequenceNumber* and the *timeStamp* they are informing about. Now we explain the sequence to synchronize the network in Fig. 1. We indicate the message sent specifying the fields: *SyncBC* (*propagatorID*; *sequenceNumber*; *timeStamp*) and *TimeUC* (*timeStamperID*; *propagatorID*; *sequenceNumber*; *timeStamp*). The first hop would be as follows:

1) N6 initiates the synchronization process by sending a *SyncBC* (*N6*; *0*; *void*) message. The nodes that receive the message (N1, N2...) timestamp the arriving time of the SyncBC from node 6 with sequence number 0; that is to say: $TS_{N1}\{N6, 0\}$, $TS_{N2}\{N6, 0\}$...
2) N1 can inform N6 about the timestamp when it receives the last SyncBC message: *TimeUC* (*N1*; *N6*; *0*; $TS_{N1}\{N6, 0\}$).
3) N6 sends the timestamp of the previous SyncBC message and sets a new reference point for time-stamping: *SyncBC* (*N6*; *1*; $TS_{N1}\{N6, 0\}$). At this instant, each N*i* neighbour of N6 has its respective sync-points from the first SyncBC: [$TS_{Ni}\{N6, 0\}$, $TS_{N1}\{N6, 0\}$]. N1 does not need it because it rules the GNT.

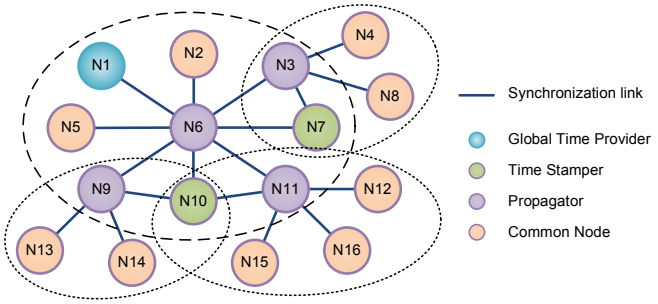

Fig. 1. Grid Network. Propagator nodes send broadcast messages, and time stamper nodes reply to them with the arrival time of the broadcast. All the nodes have also direct communication links with their neighbors.

From now on, steps 2 and 3 will be repeated causing all nodes in range of N6 to have a collection of sync-points. Then, using linear regression, they get synchronized; calculating their offset and slope differences to the reference clock. In that first hop, all nodes within the dashed ring will be synchronized to N1's clock.

Note that N6 does not have the global time. To fix this and to propagate the clock one hop away, any other propagator, such as N3, initiates the above described sequence. The subsequent hops will be performed following the same procedure. Each propagator broadcasts SyncBC messages including the timestamp of the previous SyncBC message. Time-stampers notify the propagators about the last timestamp. All nodes in a range get a collection of sync-points that allow them to synchronize to GNT. Accuracy of the sync-points will be degraded as nodes are farther away from the clock generator (N1), similar to sender—receiver methods. When synchronizing the nodes situated within the dotted rings, the time-stampers used by the propagators (N7 and N10) are one-hop away from the GNT (N1), which will increase the error committed.

Once the network is synchronized, it can support the mobility, connection and disconnection of nodes without synchronization duties. In case of near failure (i.e. low batteries) propagators and time-stampers can transfer their responsibilities to neighboring nodes. In any case, similar to the scheme proposed by Maróti et al., a network could be autonomous and self-healing in terms of synchronization using node identifiers to automatically assign roles [11].

III. MBS PROTOCOL EVALUATION AND COMPARISON

We have evaluated the MBS scheme in two different architectures. Both are used to build multi-hop wireless sensor networks following two standard protocols: ZigBee and Bluetooth [13]. Regarding to the hardware, the method used is similar in both cases; a microcontroller that manages a communications transceiver. In the case of ZigBee, we used a common platform: an Atmel microcontroller (ATMega128) with the Chipcon's CC2420 transceiver [12]. Developing the entire stack to make the network ZigBee-compliant requires a lot of work, thus we used the EmberZNet embedded software [14]. This way, we built a multi-hop, auto-routing and self-healing ZigBee network working at the application layer.

The architecture used to implement the Bluetooth network is similar to the one presented by Beutel [13]. A Microchip PIC16F876 microcontroller manages a Mitsumi Bluetooth module (WML-C20) through HCI, the standard Bluetooth Host Controller Interface. This design enables us to access low-power modes and implement tree topology networks using scatternets.

In both cases, time-stamping at the sender's side is not considered by the application programming interface (API) used. Thus, we can only use reference broadcasts to synchronize nodes.

A. Single-hop Synchronization

When synchronizing wireless nodes, all methods use one of the following strategies: (a) to timestamp at the sender and receiver side, or (b) to use reference broadcast time-stamping only at the arriving side. In Table 1, we compared the alignment errors of some synchronization schemes presented in bibliography.

The timing accuracy among nodes depends mainly on the hardware and firmware architecture: how the sending and arriving moments are detected, and which times (propagation, access, etc.) are affected and their uncertainty. Errors showed in Table 1 determine the accuracy of each sync-point that will be used to perform the linear regression in (1). Of course, the less the errors are, the more accurate the synchronization will be. Other factors that also affect the estimation are:

- The distribution of the errors (uniform, Gaussian, etc.) will determine how the linear regression eliminates them and the quality of the clock estimation. We have found out that the time difference between reception instants of broadcast messages follows a Gaussian distribution with the architectures described before (Fig. 2 shows the histogram using Bluetooth). This result agrees with that

obtained in [6] with the Berkley Motes.

- The local oscillator drift is influenced by the initial accuracy (difference between the oscillator output frequency and the specified frequency at 25°C at the time of shipment by the manufacturer), temperature stability and aging. Its behavior can also condition the precision and the timing lifetime.
- Finally, the frequency and number of sync-points used in the estimation will determine the expected accuracy.

As stated in by van Greunen and Rabaey, not all the sensor networks applications have the same synch needs in terms of accuracy [15]. In order to provide the reader with some guidelines that could help in deciding the best suited hardware (transceivers, crystals, etc.) and firmware (sync-messages rate, and number of data points to perform regression) in each application, we have characterized synchronization behavior in several scenarios. We compared the two architectures described above, Bluetooth and ZigBee, each with different alignment errors as shown in Table 1. Both alternatives have been tested with two local oscillators with different frequency stabilities of 40 PPM and 150 PPM. We have also changed the synchronization rates (30 s and 300 s) and number of sync-points used in the regression (3, 6, 9, 12, 20 and 50).

TABLE 1
COMPARISON OF ALIGNMENT ERROR IN SYNCHRONIZATION METHODS

		Average error (µs)	Worst case error (µs)
Sender — Receiver synchronization			
Zigbee	(Motes 2.4 GHz) [13]	14.9	61.0
TPSN	(Motes 916 MHz) [10]	16.9	44.0
FTSP	(Motes 433 MHz) [11]	1.4	4.2
Receiver — Receiver synchronization			
RBS	(Motes) [6]	21.9	93.0
MBS	(Bluetooth)	4.5	18.0
MBS	(ZigBee)	22.2	52.0

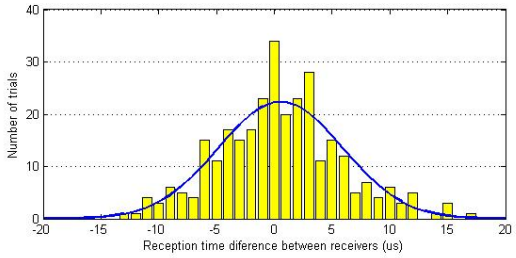

Fig. 2. Histogram of reception time difference between Bluetooth receivers.

In the test we performed, we synchronized five nodes. One of them periodically broadcasts a pulse over a wire in order to minimize timing errors. As the remainder of the nodes in the network are continuously polling the arrival of pulses, differences in the receiving instant among nodes is far below the synchronization error. On every detection of the pulse, nodes save their estimated global time that is collected afterwards. In Table 2 we show the average and maximum synchronization error with 95% probability in the scenarios described above.

The first interesting result is that synchronization precision depends mainly on the number and accuracy of the sync-points used to estimate parameters by regression. Thus, if we have very low computation resources or if we need low accuracy timing we can use few sync-points; i.e. a lightweight method [15]. In contrast, if highest accuracy is needed, then we will need powerful hardware to use the maximum number of points in the regression.

Contrary to what might some people expect and agreeing with what Maróti et al have found out, synchronization rate and oscillator's accuracy barely affect precision [11]. This makes sense if we consider the local clock stable at short-medium term, something that fortunately will happen in most cases. Aging has a negligible effect on stability: one to three PPM each year. Temperature influences clock's drift more severely: tens of PPMs in the operating temperature range. In worst cases, this translates into an error of one microsecond each º C. In most applications this will not be considered a problem, because temperature will not vary drastically and the sync-points' updating will automatically compensate this drift. Anyhow, temperature sensors could be used to detect big gradients and increase the rate of synchronization messages compensating the errors.

TABLE 2
SYNCHRONIZATION ERROR IN MBS METHOD WITH ONE HOP

	ZigBee 40 PPM	ZigBee 150 PPM	Bluetooth 40 PPM	Bluetooth 150 PPM
N / T [a]	Avg. / Max. [b]	Avg. / Max. [b]	Avg. / Max. [b]	Avg. / Max. [b]
3 / 30	39.26 / 105.20	39.45 / 106.42	7.67 / 20.92	8.10 / 21.70
3 / 300	39.99 / 108.09	39.59 / 109.05	8.00 / 21.61	7.73 / 20.88
6 / 30	21.12 / 52.65	22.06 / 55.23	4.17 / 10.30	4.34 / 10.77
6 / 300	21.90 / 53.52	21.29 / 52.07	4.41 / 10.99	4.36 / 10.93
9 / 30	17.27 / 40.95	16.58 / 40.66	3.30 / 8.19	3.30 / 8.25
9 / 300	16.10 / 39.38	15.72 / 38.90	3.28 / 8.22	3.36 / 8.39
12 / 30	13.33 / 32.83	14.39 / 35.13	2.82 / 6.84	2.79 / 6.89
12 / 300	13.99 / 34.52	13.77 / 34.87	2.73 / 6.62	2.68 / 6.63
20 / 30	11.29 / 27.20	9.83 / 24.23	1.96 / 4.71	2.13 / 5.14
20 / 300	10.61 / 26.13	10.46 / 25.51	2.19 / 5.29	2.13 / 5.30
50 / 30	7.03 / 17.46	6.77 / 15.58	1.23 / 2.93	1.27 / 3.10
50 / 300	6.22 / 14.80	7.54 / 18.43	1.30 / 3.27	1.32 / 3.21

[a] N is the number of sync-points used to perform regression and T is the synchronization interval in seconds.
[b] Average and Maximum synchronization error with 95% probability in µs.

The reduced influence of synchronization rate can be used in many ways. By increasing it, we can reduce the initial settling time in multi-hop networks, quickly synchronize a new node need or maintain the accuracy in sudden temperature variations. Lowering the rate will be useful to minimize extra traffic and reduce power consumption. Despite the results showed in Table 2, the synchronization interval cannot be as long as we want. Linear regression estimates the skew and the offset of the local clock referred to the GNT, and the more time from the last resynchronization, the larger the error of this estimation.

B. Multi-hop GNT Propagation

Cumulative errors when propagating the network time only depend on the estimation's accuracy of the corresponding time-stamper's clock. When the time-stamper is the same setting the reference, the precision will be the one described in the previous section. Differences arise when it owns an n-hop estimation of the network clock.

As in the single-hop case, we have evaluated the behavior of MBS in two four-hop depth networks implemented with ZigBee and Bluetooth, and tested the performance with different number of sync-points. Synchronization errors for both scenarios are showed in Fig. 3.

As in the case of one hop, synchronization error depends chiefly on the number of points used to perform regression and on the alignment error in the estimation of sync-points. We can see how as the number of hops increases, synchronization error becomes sensitive to the number of sync-points used. Again, if the application needs higher synchronization accuracy than the alignment error between nodes, it is possible to reduce error increasing the number of pairs to perform regression.

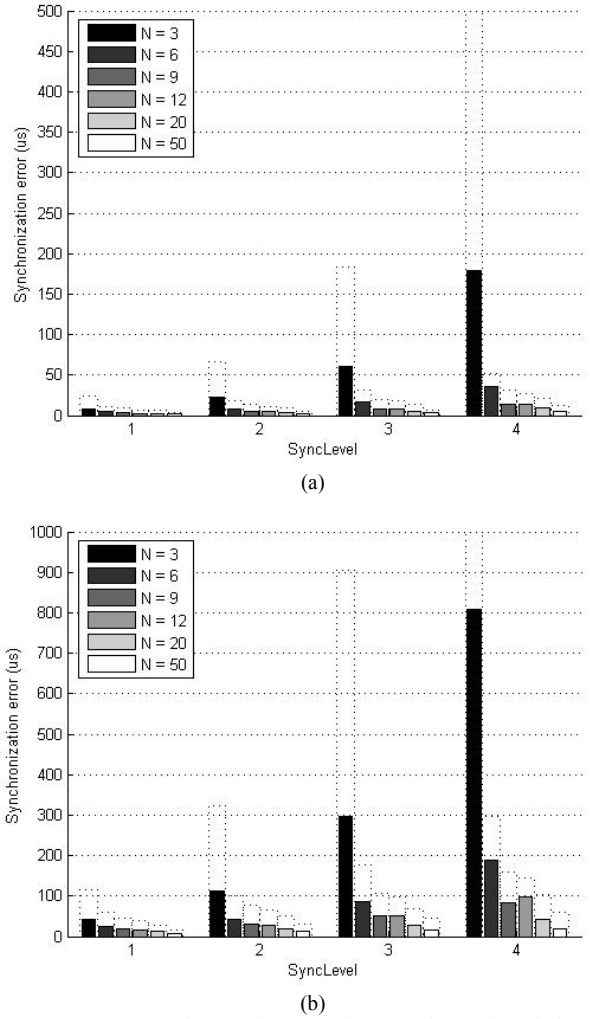

Fig. 3. Average synchronization error for (a) Bluetooth and (b) Zigbee networks with 4 hops and sync messages every 30 seconds. N is the number of points used to perform regression. Dashed bars represent maximum error with 95% probability.

We find no point in comparing precision among methods because it mainly depends on the accuracy estimating sync-points and on the number of pairs used. That is to say, architecture (communication transceiver, protocol, memory available, etc.) will be much more relevant than the synchronization protocol used. On the other hand, the amount of messages needed, will have a big influence on the applicability of the method. This is just one of the strongest points of MBS; it drastically reduces the number of messages compared to other receiver—receiver protocols [6, 8]. Indeed, it is in the same order of magnitude than the FTSP (the most efficient sender—receiver method) [11].

IV. CONCLUSIONS

Synchronization methods, which work at the MAC layer get rid of the non-deterministic error due to the medium access time by accurately time-stamping transmission instants, however, this is not possible when working at the highest layers of standard protocol stacks. In these cases, receiver-receiver methods may be used, but existing methods set a network time shared by all the nodes (including the broadcast senders) at the expense of a high network load; inadmissible for large networks. In this paper, we have presented MBS, a multi-hop broadcast synchronization protocol that is able to efficiently set a common global time in wireless and satellite networks. The key issue of the technique is that each reference broadcast informs about the timestamp of the previous one. This way, message exchanging is minimized; being similar to other sender-receiver methods.

We have implemented MBS protocol in two different architectures (Bluetooth and ZigBee) and evaluated its performance. Through an extensive set of experiments, we have determined the factors that influence synchronization accuracy.

REFERENCES

[1] D. Niculescu, Communication paradigms for sensor networks, *IEEE Communications Magazine*. 43 (3), (2005) 116-122.

[2] I. F. Akyildiz *et al*., A survey on sensor networks, *IEEE Communications Magazine*. 40 (8), (2002) 102-114.

[3] F. Sivrikaya and B. Yener, Time Synchronization in Sensor Networks: A Survey, *IEEE Network*. 18 (4), (2004) 45-50.

[4] P. Ubolkosold, S. Knedlik, O. Loffeld, Clock synchronization protocol for distributed satellite networks. Proceedings. IEEE International Geoscience and Remote Sensing Symposium, 2005. 25-29.

[5]B. Sundararaman, U. Buy and A.D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, *Ad Hoc Networks.* 3 (2005) 281-323.

[6] J. Elson, L. Girod and D. Estrin. Fine-Grained Time Synchronization using Reference Broadcasts, *Proc. 5th Symp. Op. Sys. Design and Implementation*, (Boston, MA, 2002) 147-163.

[7] J. Elson, D. Estrin, Time Synchronization for Wireless Sensor Networks. *Int'l. Parallel and Distrib. Processing Symp.,* (San Francisco, CA, 2001).

[8] S. PalChaudhuri, A. Saha, D.B. Johnson, Adaptive Clock Synchronization in Sensor Networks, *Proc. 3rd IEEE Information Processing in Sensor Networks (IPSN)*, (Berkeley, CA, 2004).

[9] Y.W. Hong and A. Scaglione, A Scalable Synchronization Protocol for Large Scale Sensor Networks and Its Applications, *IEEE Journal on Selected Areas in Communications (JSAC)*. 23 (5), (2005) 1085-1099

[10] S. Ganeriwal, R. Kumar and M.B. Srivastava, Timing-sync protocol for sensor networks, *ACM SENSYS*, (Los Angeles, CA, 2003).

[11] M. Maróti, B. Kusy, G. Simon and A. Lédeczi, The flooding time synchronization protocol. *SenSys '04:* 39-49.

[12] D. Cox, A. Milenkovic and E. Jovanov, Time Synchronization for ZigBee Networks, *Proc. of the 37th Southeastern Symposium on System Theory (SSST2005)*, (Tuskegee, AL, 2005) 135-138

[13]http://www.ember.com/products/ software/index.html.

[14] J. Beutel, Robust Topology Formation using BTnodes, *Computer Communications*. 28 (13), (2005) 1523-1530.

[15] J. van Greunen and J. Rabaey, J, Lightweight time synchronization for sensor networks. *Proc. of the 2nd ACM Int. Conference on Wireless Sensor Networks and Applications*, (San Diego, CA, 2003) 11-19.