A distributed consensus protocol for clock synchronization in wireless sensor network

Luca Schenato, Giovanni Gamba

***Abstract*— This paper describes a novel consensus-based protocol, referred as *Average TimeSync* (ATS), for synchronizing a wireless sensor. This algorithm is based on a class of popular distributed algorithms known as consensus, agreement, gossip or rendezvous whose main idea is averaging local information. The proposed algorithm has three main features. Firstly, it is fully distributed and therefore robust to node failure and to new node appearance. Secondly, it compensates for clock skew differences among nodes, thus maintaining the network synchronized for longer periods than using simple clock offset compensation. Finally, it is computationally lite as it involves only simple sum/product operations. The algorithm has been implemented and preliminary experimental results are provided.**

I. INTRODUCTION

Recent advances in technology have made low cost, low power wireless sensors a reality. For several applications of a wireless sensor networks, such as mobile target tracking, event detection, efficient TDMA scheduling, and sleep scheduling with very low duty cycle, it is essential that the nodes act in a coordinated and synchronized fashion. All these applications require a global clock synchronization, that is all the nodes of the network need to refer to a common notion of time. In fact, for example, let us consider the problem of tracking a moving target using proximity sensors, where some nodes are deployed in the environment and their proximity sensors detect when the moving object passes in their vicinity [13]. Assuming that the position of the sensors is known, it is essential that the instants of detection are precisely time-stamped for determining the trajectory (direction and speed) of the moving object. Clearly, the precision of the tracking algorithm based on this system is limited by the accuracy of the clock synchronization. Another important application needing a time-synchronization service, is habitat monitoring [16] [17], where many battery-powered nodes are deployed in the area of interest for an extended period of time without possibility of battery replacement. In this scenario the nodes have to operate with very low duty cycle, i.e. they need to be turned off for most of the time since listening the channel for receiving a transmission is the most power consuming operation in low-power wireless nodes (see for example [11], pag.14). Therefore, it is necessary to design a time schedule in which *all* nodes turn on their radios at designated times, transmit the data they have recorded to the base station via multi-hop, and then turn off the radio again. If nodes are awake at different times, the sensor network might not be fully connected and therefore packets might not be correctly delivered to the base station. Therefore, time synchronization accuracy is directly related to power consumption, since it is necessary to increase the duration of the awake intervals, i.e. the intervals when the radios are turned on an expecting to receive a transmission, to compensate for synchronization errors. First, let us consider the problem of synchronizing two clocks. This is generally solved by selecting one clock and synchronizing it relative to the other clock. Synchronization is achieved by adding to the current local time of one clock the instantaneous time difference with the other clock. However, due to small fabrication variations or ambient conditions, clocks have slightly different skews, i.e. they run at different speeds (see solid lines in Fig. 1(left)), therefore if no skew compensation is adopted, the offset resetting between two clocks needs to be performed quite often, as shown in Fig. 1(right). To avoid frequent offset resetting, many synchronization algorithms adopt skew compensations techniques, i.e. they estimate how fast the node to be synchronized is running with respect to the reference node and then they use this to compensate the synchronizing clock reading, as shown in the dashed line in Fig. 1(left). By combining skew compensation and offset resetting it is possible to maintain two clocks synchronized for long periods of time.

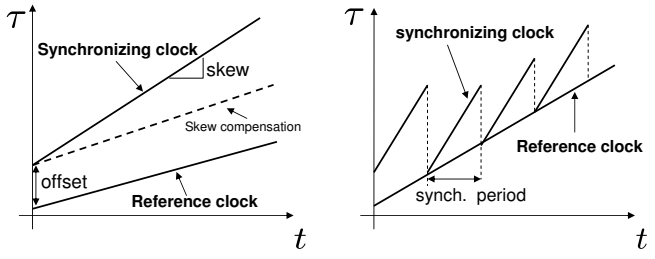

Fig. 1. Local time, τ, of two nodes with different clock skews relative to absolute time, t. The dashed line correspond to synchronizing clock after skew compensation relative to the reference clock (*left*). Periodic offset synchronization of one clock relative a reference clock (*right*).

Time-synchronization of a wireless sensor network adds two major problems to the previous discussion. The first problem emerges from the fact that in a sensor network the nodes cannot communicate directly with each other but they have to do it via multi-hop. Therefore, it is not possible to choose a reference node to which all other nodes can be

L. Schenato is with Faculty of Department o Information Engineering, University of Padova, Italy schenato@dei.unipd.it

G. Gamba is a Ph.D. candidate at the Department o Information Engineering, University of Padova, Italy giogamba@dei.unipd.it

This work has been partially supported by European Union project SENSNET founded by Marie Curie IRG grant n.014815

synchronized to. The second problem is a consequence of the unpredictable time delay between the clock reading in one node and its processing at the receiver node, thus causing poor performance. In fact, delivery time of radio messages in WSNs are subject to random variations due to many factors, such as interference, backoff due to occupied radio channel, and scheduling of the host operating system of the motes. These delays can be magnitudes larger than the required precision of time synchronization. Different strategies has been proposed to solve these two problem. The next section reviews some of the most important ones.

II. Previous work

A. Communication topology

One natural approach to deal with the multi-hop nature of a sensor network is to organize the network in a rooted tree as suggested in the Time-synchronization Protocol for Sensor Networks (TPSN) proposed by Ganeriwal *et al.* [7]: initially one node is elected to be the clock reference, then a spanning tree rooted at that node is builded, and the offset of any node with respect to the root is obtained simply by adding the offset of the edges in the unique path from each node to the root, as graphically shown in Fig.2(center).

The offset between two nodes (in adjacent levels of the tree) is calculated with a two-way message exchange, which bounds the transmit and propagation delay. This approach suffers from two limitations. The first limitation arises because if the root node or parent node dies, then a new root-election or parent-discovery procedure needs to be initiated, thus adding substantial overhead to the code and long periods of network de-synchronization. The second limitation is due to the fact that geographically close nodes, such as the node i and j in Fig.2(center) might be far in terms of the tree distance, which is directly related to the clocks error. This is particularly harmful for many applications such as object tracking or TDMA scheduling, for which it is really important that clock errors between one node and the other should degrade sufficiently smoothly as a function of geographic distance. More recently, a similar scheme, the Flooding Time Synchronization Protocol (FTSP, [10]) proposed to use broadcast communications and MAC layer time-stamping to achieve better precision. Topologically, it is similar to the TPSN approach since it builds a rooted tree, but it implements also mechanisms for skew compensation, dynamic topology adaptation and root failure recovery. These mechanisms increase performance and robustness, but still do not completely solve the issues aforementioned.

Another approach to the same problem is to divide the network in interconnected single-hop clusters, as suggested in the Reference Broadcast Synchronization (RBS) scheme [4] and graphically illustrated in Fig.2(right). In this protocol, within every cluster a reference node is selected to synchronize all the other nodes. The reference nodes of different clusters are synchronized together and acts as gateways by converting local clocks of one cluster into local clocks of another cluster when needed. As the TPSN, also RBS suffers from large overhead necessary to divide the network into clusters and to elect the reference nodes, and it is fragile to node failures.

The last approach is have a fully distributed communication topology where there are no special nodes such as roots or gateways, and all nodes run exactly the same algorithm. This approach has the advantage to be very robust to node failure and new node appearance, but requires novel algorithms for the synchronization as there is no reference node. One example of a completely distributed synchronization strategy is the Reachback Firefly Algorithm (RFA), inspired by firefly synchronization mechanism [19]. In this algorithm every node periodically broadcasts a synchronization message and anytime they hear a message they advance of a small quantity the phase of their internal clock that schedules the periodic message broadcasting. Eventually all nodes will advance their phase till they are all synchronized, i.e. they "fire" a message at the same time. This approach however does not compensate for clock shew, therefore the firing period needs to be rather small. Recently, Solis *et al.* [15] proposed a Distributed Time Synchronization Protocol (DTSC) which is fully distributed and compensate also for clock skews. This protocol is similar to our Average TimeSync protocol, but it is formulated as a distributed gradient descent optimization problem [8], while ours as a consensus problem. However, further investigation of similarities and differences of the two algorithms is needed.

B. Delivery time delay

As mentioned in the previous section, the other major problem to be faced in WSN clock synchronization, is the random delivery time of messages. In particular, it is possible to decompose the total delivery time into different parts, as thoroughly analyzed in [7] and [10] and pictorially shown in Fig. 3:

- *Send Time*, T_s: time needed to read the local clock, assemble the message, and do the send-request to the MAC layer on the transmitter side. Depending on the system call overhead of the OS and on the current processor load, the send time is non deterministic and can be as high as hundreds of milliseconds.
- *Access Time*, T_a: waiting time to access the channel until transmission begins. It depends on the traffic on the radio channel and the backoff time of the CDMA protocol implementation. It varies from milliseconds up to seconds depending on the current network traffic.
- *Transmission time*, T_t: time necessary for the sender to transmit the message. This time is in the order of tens of milliseconds depending on the length of the message and the speed of the radio.
- *Propagation time*, T_p: travel time of a message from sender to receiver. The propagation time is highly deterministic and it depends only on the distance between the two nodes. This time is less than one microsecond for node distances under 300 meters.
- *Reception time*, T_{rp}: time necessary for the receiver to receive the message. It is the same as the transmission time, i.e. $T_{rp} = T_t$.

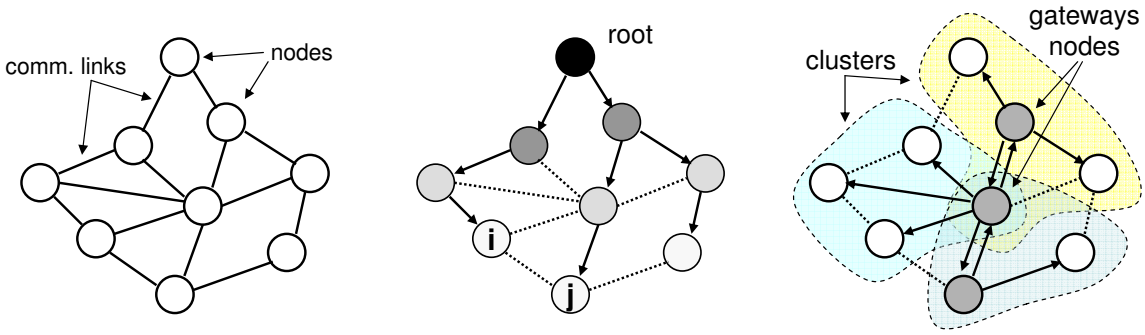

Fig. 2. Sensor network graph with available communication links (*left*). Tree-based synchronization (TPSN and FTSP): a feasible communication tree is determined and then every node synchronize with its parent. Decreasing intensity circle color indicates hops distance from root, and dotted lines represent unused communication links (*center*). Cluster-based synchronization (RBS): every node in a cluster is synchronized to its gateway node, and gateway nodes synchronize with themselves. Arrows indicates direction of clock synchronization messages. Shaded circles represent gateway nodes (*right*).

- *Receive time*, T_{rv}: time required to process the incoming message and to notify the reception to the application. It is similar to the send time.

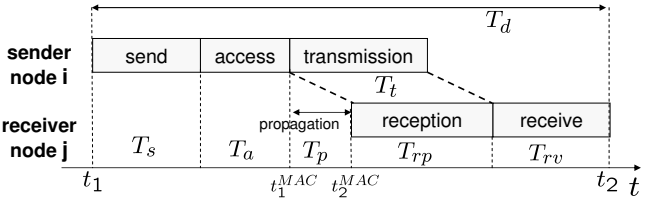

Fig. 3. Temporal illustration of packet transmission and reception delays between two wireless sensor nodes.

The total delivery delay, T_d is then given by:

$$T_d = T_s + T_a + T_p + T_{rp} + T_{rv} \quad (1)$$

which is directly related to the minimum achievable synchronization error. Delay can be compensated for as long as it can be computed accurately. *Propagation time* is negligible in WSN, as it is much smaller than the clock resolution. *Transmission time* and *Reception time* are predictable from the speed of the radio and the length of the message to transmit. However, *Access time*, *Send time* and *Receive time* are rather unpredictable, causing delay uncertainty on the order of hundreds of milliseconds.

Different strategies has been proposed to limit this problem. The simplest strategy is performed via monodirectional transmission: node i reads its clock τ_i at time t_1 and sent it to node j which reads its own local clock τ_j at time t_2 and then computes the clock offset as $o_{ji} = \tau_j(t_2) - \tau_i(t_1)$. The delivery delay $T_d = t_2 - t_1$ cannot be estimated exactly due to randomness of T_s, T_a and T_{rv}.

A more efficient strategy is obtained via bidirectional transmission by re-sending back to the sender node i a packet with clock reading of node j as soon j has received the packet from i. If the delivery delay T_d was constant on both packet transmission from node i to j and vice versa, then this strategy could compute it exactly and remove it from the clock offset estimation as $o_{ji} = \tau_j(t_2) - \tau_i(t_1) - T_d \simeq \tau_j(t_1) - \tau_i(t_1)$. This scheme has been used, for example, in the Lightweight Time Synchronization (LTS) protocol [18] and in the TPSN [7]. Unfortunately, due to the delivery delay randomness of any transmission, only a limited benefit can be obtained. Better performance can be obtained by repeating this process and averaging the offset estimate, at the price of higher communication load.

An alternative approach to avoid some randomness in the delivery delay is the broadcast-based synchronization, which adopts a beacon node to synchronize two or more other nodes within its transmission range. In this strategy a reference packet is broadcasted from the beacon node k at time t_1^k, then all nodes that receive that packet reads their local clock, i.e. $\tau_i(t_2^i) = \tau_i(t_1^k + T_s^k + T_a^k + T_p + T_t^k + T_{rv}^i)$ and $\tau_j(t_2^j) = \tau_j(t_1^k + T_s^k + T_a^k + T_p + T_t^k + T_{rv}^j)$, and finally they communicate each other their readings to compute the clocks offset as $o_{ji} = \tau_j(t_2^j) - \tau_i(t_2^i)$. This approach removes the randomness from the sender side as the delay in the clock readings of node i and j is $t_2^j - t_2^i = T_{rv}^j - T_{rv}^i$, thus giving a much better precision than the round-trip strategy described above. Nonetheless, still some uncertainty persists due to the possible difference between the receive time T_{rv} of the two nodes. This strategy has been used, for example, in the RBS protocol [4].

The most effective strategy, whenever technologically possible, is to use MAC-layer time-stamping, as proposed in the FTSP [10]. In this approach, the clock reading of sender node i is performed right after the first bit of the packet is sent by the receiver, i.e. $\tau_i(t_1^{MAC})$, and the clock reading of the receiver node j is performed right after the first bit has arrived, i.e. $\tau_j(t_2^{MAC})$, as shown in Fig.3. Since the propagation time is negligible, then we can safely assume that clock reading on the two nodes is performed instantaneously, i.e. $t_1^{MAC} \simeq t_2^{MAC}$, thus removing the randomness of the packet delivery time from both sender and receiver side.

The Average TimeSync protocol proposed in this paper is fully distributed, i.e. does not require any special root, includes skew compensation, and exploits MAC-layer time-

	distrib.	skew comp.	MAC timestamp
TPSN [7]	no	no	no
LTS [18]	no	no	no
FTSP [10]	no	yes	yes
RBS [4]	no	yes	yes
RFA [19]	yes	no	yes
DTSP [15]	yes	yes	yes
ATS	yes	yes	yes

TABLE I
TIME-SYNCHRONIZATION FEATURES FOR DIFFERENT PROTOCOLS.

stamping for higher accuracy. Besides the DTSP, none of today's available protocols possesses all these features, as summarized in Table I.

III. MODELING

In this section, we provide a mathematical modeling for wireless sensor network clocks. Every node i in a WSN has its own local clock whose first order dynamics is given by:

$$\tau_i(t) = \alpha_i t + \beta_i \tag{2}$$

where τ_i is the local clock reading, α_i is the local clock skew which determine the clock speed, and β_i is the local clock offset. Since the absolute reference time t is not available to

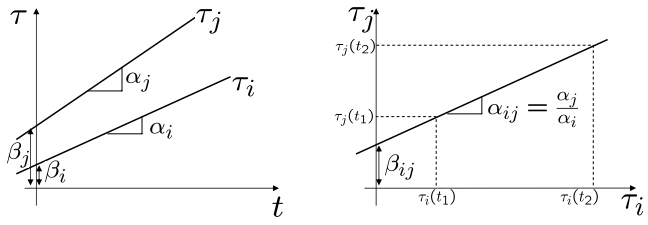

Fig. 4. Clocks dynamics as a function of absolute time t *(left)*, and relative to each other *(right)*.

the nodes, it is not possible to compute the parameters α_i and β_i. However, it is still possible to obtain indirect information about them by measuring the local clock of one node i with respect to another clock j. In fact, if we solve Eqn. (2) for t, i.e. $t = \frac{\tau_i - \beta_i}{\alpha_i}$ and we substitute it into the same equation for node j we get:

$$\begin{aligned} \tau_j &= \frac{\alpha_j}{\alpha_i}\tau_i + (\beta_j - \frac{\alpha_j}{\alpha_i}\beta_i) \\ &= \alpha_{ij}\,\tau_i + \beta_{ij} \end{aligned} \tag{3}$$

which is still linear as shown in Fig. 4(right). We want to synchronize all the nodes with respect to a *virtual reference clock*, namely:

$$\tau_v(t) = \alpha_v t + \beta_v \tag{4}$$

Every local clock keeps an estimate of the virtual time using a linear function of its own local clock:

$$\hat{\tau}_i = \hat{\alpha}_i \tau_i + \hat{o}_i \tag{5}$$

Our goal is to find $(\hat{\alpha}_i, \hat{o}_i)$ for every node in the WSN such that:

$$\lim_{t\to\infty} \hat{\tau}_i(t) = \tau_v(t), \qquad i = 1, \ldots, N \tag{6}$$

where N is the total number of nodes in the WSN. Therefore, if the previous expression is satisfied, then all nodes will have a common global reference time given by the virtual clock time. The previous expression can be rewritten differently by first substituting Eqn.(2) into Eqn.(5) to get:

$$\hat{\tau}_i(t) = \hat{\alpha}_i \alpha_i t + \hat{\alpha}_i \beta_i + \hat{o}_i \tag{7}$$

therefore Eqn. (6) is equivalent to:

$$\lim_{t\to\infty} \quad \hat{\alpha}_i(t) \;= \frac{\alpha_v}{\alpha_i}, \tag{8}$$

$$\lim_{t\to\infty} \quad \hat{o}_i(t) \;= \beta_v - \hat{\alpha}_i(t)\beta_i = \beta_v - \frac{\alpha_v}{\alpha_i}\beta_i, \quad i = 1, .., N \tag{9}$$

Before moving to the next section which present how the ATS protocol updates $(\hat{\alpha}_i, \hat{o}_i)$ to satisfy the previous expression, it is important to remark few points. The first regards the clock modeling of Eqn.(4). In reality the parameters $\alpha_i(t), \beta_i(t)$ are time varying due to ambient conditions or aging, therefore the updating period of the synchronization protocol should be shorter than the variations of these parameters.

The second point is that the virtual reference clock it is a fictitious clock and it not fixed a priori. In fact, the values of its parameters (α_v, β_v) are not important, since what it is really relevant is that all clocks converge to *one* common virtual reference clock. Indeed, as it will be shown in the next section, the parameters (α_v, β_v) to which the local clock estimates converge depend on the initial condition and the communication topology of the WSN.

The last remark is that by using MAC-layer time-stamping, we can safely assume that the reading of the local clock $\tau_i(t_1)$, packet transmission and reading of the local clock $\tau_j(t_2)$ is instantaneous, i.e. $t_1 = t_2$. If this not the case, our synchronization protocol cannot be used as it is and needs to be modified to cope with packet delivery delay. However, this does not seem very harmful as most of new generation sensor network nodes, such as the Tmote Sky mote [11], include MAC-layer time-stamping. In fact, it is widely adopted in all recently proposed protocols as indicated in Table I.

IV. THE ATS PROTOCOL

The Average TimeSync protocol includes three main parts: the relative skew estimation, the skew compensation, and the offset compensation.

A. Relative Skew Estimation

This part of the protocol is concerned with deriving an algorithm to estimate for each clock i the relative skew with respect its neighbors j. Let N_i the number of nodes that can transmit packets to node i with a single hop. Every node i tries to estimate the relative skews $\alpha_{ij} = \frac{\alpha_j}{\alpha_i}$ with respect to its neighbor nodes j. This is accomplished by storing the current local time $\tau_j(t_1)$ of node j into a broadcast packet, then the node i that receives this packet immediately

record its own local time $\tau_i(t_1)$. As discussed in the previous section, we can assume that the readings of the two local clocks is instantaneous since we are using MAC-layer time-stamping. Therefore, node i records in its memory the pair $\big(\tau_i(t_1), \tau_j(t_1)\big)$. When a new packet from node j arrives to node i, the same procedure is applied to get the new pair $\big(\tau_i(t_2), \tau_j(t_2)\big)$, as shown in Fig.4(right), and the estimate of the relative skew α_{ij} is performed as follows:

$$\eta_{ij}^{+} = \rho_\eta \eta_{ij} + (1-\rho_\eta)\frac{\tau_j(t_2)-\tau_j(t_1)}{\tau_i(t_2)-\tau_i(t_1)} \tag{10}$$

where the symbol η_{ij}^{+} indicates the new value assumed by the variable η_{ij}, and $\rho_\eta \in (0,1)$ is a tuning parameter. If there is no measurement error and the skew is constant, then the variable η_{ij} converges to the variable α_{ij} as stated in the following proposition:

Proposition 1: Let us consider the update Equation (10) where $0 < \rho_\eta < 0$ and each τ_i evolves according to Equation (2). Then

$$\lim_{k\to\infty} \eta_{ij}(t_k) = \alpha_{ij} \tag{11}$$

for any initial condition $\eta_{ij}(0) = \eta(0)$, where t_k indicates the update instants.

Proof: The proof follows easily from the fact that $\frac{\tau_j(t_2)-\tau_j(t_1)}{\tau_i(t_2)-\tau_i(t_1)} = \alpha_{ij}$ regardless of the two sampling instants t_1 and t_2. In fact we have that

$$\eta_{ij}(t_k) = \rho_\eta^k \eta(0) + \sum_{l=1}^{k-1}(1-\rho_\eta)^l \alpha_{ij} = \rho_\eta^k \eta(0) + \alpha_{ij}(1-\rho_\eta^k)$$

Since $0 < \rho_\eta < 1$, if we take the limit for $k \to \infty$, we obtain the claim. ∎

In practise, Eqn. (10) acts a low pass filter where the parameter ρ_η is used to tune the trade-off between the speed of convergence (ρ_η close to zero) and noise immunity (ρ_η close to unity). In fact, filtering is necessary because the quantity $\frac{\tau_j(t_2)-\tau_j(t_1)}{\tau_i(t_2)-\tau_i(t_1)}$ is not constant but it is slowly time-varying and affected by quantization noise. It is important to remark that it is not necessary to perform the update at a fixed frequency, i.e. the packet inter-arrival $t_2 - t_1$ can vary, thus making this algorithm particularly useful for asynchronous communication. The other important advantage of this algorithm is that it requires little memory. In fact, each node i needs to store only the current N_i relative skew estimates η_{ij} and the last local clock pairs $\big(\tau_i(t_1), \tau_j(t_1)\big)$ recorded. Since the parameter N_i is in general small even for large networks, this algorithm is also rather scalable.

B. Skew Compensation

This part of the algorithm is the core of the Average TimeSync protocol, as it forces all the nodes to converge to a common virtual clock rate, α_v, as defined in Eqn. (4). The main idea is to use a distributed consensus algorithm based only on local information exchange. In consensus algorithms any node keeps its own estimate of a global variable, and it updates its value by averaging it relative to the estimate of its neighbors [12]. In practice, every node bootstraps each other till all of them converge to a common value, i.e. till they agree upon a global value. The algorithm is very simple, in fact every node stores its own virtual clock skew estimate $\hat{\alpha}_i$, defined in Eqn. (5). As soon as it receives a packet from node j, it updates $\hat{\alpha}_i$ as follows:

$$\hat{\alpha}_i^{+} = \rho_v \hat{\alpha}_i + (1-\rho_v)\eta_{ij}\,\hat{\alpha}_j \tag{12}$$

where $\hat{\alpha}_j$ is the virtual clock skew estimate of the neighbor node j. The initial condition for the virtual clock skews of all nodes are set to $\hat{\alpha}_i(0) = 1$. According to Eqn. (8), we need to shown that the previous equation leads to $\lim_{t\to\infty} \hat{\alpha}_i = \frac{\alpha_v}{\alpha_i}$, which is equivalent to $\lim_{t\to\infty} \hat{\alpha}_i \alpha_i = \alpha_v$. To do so, we define the useful variable $x_i = \hat{\alpha}_i \alpha_i$. If we assume that Eqn. (11) holds, then after an initial transient period, we can assume that $\eta_{ij} = \alpha_{ij}$, therefore if we multiply all terms in Eqn. (12) by α_i, then it can be written as:

$$x_i^{+} = \rho_v x_i + (1-\rho_v) x_j \tag{13}$$

If we define the vectors $x = (x_1, x_2, \ldots, x_N)^T$ and $\mathbf{1} = (1, 1, \ldots, 1)^T$, then the previous equation can be written in matrix form as follows:

$$\mathbf{x}^{+} = A\mathbf{x} \tag{14}$$

where the matrix $A \in \mathbb{R}^{N\times N}$ has all ones on the diagonal and zeros elsewhere, except for the i-th row where $A_{ij} = 1-\rho_v$ and $A_{ii} = \rho_v$. The rows of the matrix with the ones on the diagonals, correspond to the nodes that have not received any message and therefore their $\hat{\alpha}$ are not updated. This matrix satisfies $A_{ij} \geq 0$ and $A\mathbf{1} = \mathbf{1}$. Such a matrix is called *stochastic matrix* and it has some important properties [12] [1]. The matrix A is time-varying since it depends on which nodes are exchanging synchronization messages. In fact, let us consider the product of matrices $A_T = A_{t_n} \cdots A_{t_2} A_{t_1}$, where t_i are the instants where a message is received by a node. The question is whether the product of those matrices converge, i.e. if $\prod_{n=1}^{\infty} A_{t_n} = A_\infty$. This is a rather old problem initially studied in the contest of Markov chains, which has been recently reconsidered in the context of flocking and distributed consensus [12] [14] [1]. We address the interested reader to the papers [3] [2] and references therein for a summary of many results related to necessary and sufficient conditions for convergence. Some of those results are used to prove convergence of the proposed skew compensation mechanism in the following theorem:

Theorem 1: Consider the skew update equation given by Equation (12) with initial condition $\hat{\alpha}_i(0) = 1$ and $0 < \rho_v < 1$. Also assume that $\eta_{ij} = \alpha_{ij}$ for all i, j. Also suppose that there exist $T > 0$ such that for any time window of lenght T each node i in the network transmit at least once to their neighbors their local skew compensation parameter $\hat{\alpha}_i$. Finally, let the undeline communication graph of whole sensor network is strongly connected, i.e. there is a communication path from any node i to any other node j in the network. Then

$$\lim_{t\to\infty} \hat{\alpha}_i(t)\alpha_i = \alpha_v, \quad \forall i$$

exponentially fast, where α_v is a constant parameter which satisfy the condition $\min_i(\alpha_i(0)) \leq \alpha_v \leq \max_i(\alpha_i(0))$.

Proof: Let us consider the time sequence $t_k = kT$. Let $x_i(t) = \alpha_i \hat{\alpha}_i(t)$ and the vectors $x = (x_1, x_2, \ldots, x_N)^T$ and $\mathbf{1} = (1, 1, \ldots, 1)^T$, where N is the number of nodes in the network. Let τ_h^k the district ordered communication instants of all the nodes in the network within the time window $t_k \leq t < t_{k+1}$, i.e $t_k \leq \tau_1^k < \tau_2^k < \ldots < \tau_{H_k}^k < t_{k+1}$ where H_k is the total number of communications within the window. Let $j_{\tau_{k,h}}$ which trasmits its $\hat{\alpha}_j$ at the time instact τ_h^k and assume that all its neigbors which received the message update their $\hat{\alpha}_i$ according to Equation (12). Then we can define the matrix $A_{\tau_{k,h}}$ such that all extries are zeros except for $[A_{\tau_{k,h}}]_{i,j_{\tau_{k,h}}} = 1 - \rho_v$ and $[A_{\tau_{k,h}}]_{i,i} = \rho_v$ where i is a neighbor of $j_{\tau_{k,h}}$, i.e $i \in \mathcal{N}\big(j_{\tau_{k,h}}\big)$. Such a matrix is a stochastic matrix, i.e. $A_{\tau_{k,h}}\mathbf{1} = \mathbf{1}$. According to this terminology, then we have:

$$x(t_{k+1}) = A_{\tau_{k,H_k}} A_{\tau_{k,H_k-1}} \cdots A_{\tau_{k,2}} A_{\tau_{k,1}} x(t_k) = A_k x(t_k)$$

Since each nodes communicates at least once in the k-th time window $t_k \leq t < t_{k+1}$, then the graph associated to A_k is strongly connected and therefore also rooted[1] at some node v independent of k [3]. Let now the new time sequence $\bar{t}_l = lNT$, then

$$x(\bar{t}_{l+1}) = A_{lN+N-1} \cdots A_{lN+1} A_{lN} x(\bar{t}) = \overline{A}_l x(\bar{t})$$

According to Proposition 3 (assertion 2) in [3] then graph corresponding to the matrices $\overline{A}_l$ are all strongly rooted[2] at the same node v, which is a sufficient condition to ensure that

$$\lim_{t\to\infty} x(t) = \lim_{l\to\infty} x(t_l) = \prod_{m=1}^{\infty} \overline{A}_m x(0) = x_{ss}\mathbf{1}$$

where $x_{ss} \in \mathbb{R}$. Since all $\overline{A}_m$ are stochastic, then $\max(\overline{A}_m x) \leq \max(x)$ and $\min(\overline{A}_m x) \geq \min(x)$, from which it follows that $\min(x(0)) \leq x_{ss} \leq \max(x(0))$. By recalling that $x_i(t) = \alpha\hat{\alpha}_i(t)$ and $x_i(0) = \alpha_i\hat{\alpha}_i(0) = \alpha_i$, we prove the statement of the theorem. ■

There are some important remarks about the previous theorem. The first remark is that it not important the order with which the nodes transmit, as long as they transmit from time to time. Nor it is important the exact time instants when they transmit. This implies that the protocol is totally asynchronous and that nodes can transmit with different rates. The only important condition is that the graph is sufficiently connected. In fact, the previous theorem could be relaxed to the case for which the underlying communication graph is only rooted at some node v and not strongly connected.

Another important observation is that some messages can be lost during transmission, nonetheless the conditions on rooted and strongly rooted graphs used in the theorem to prove convergence can still be guaranteed, which implies that the algorithm is robust also to link failure and packet collision.

[1]A graph is defined rooted if there exists a node v from which there is a communication path to any other node in the network

[2]A graph is defined rooted if there exists a node v from which there is a communication path of length 1 to any other node in the network.

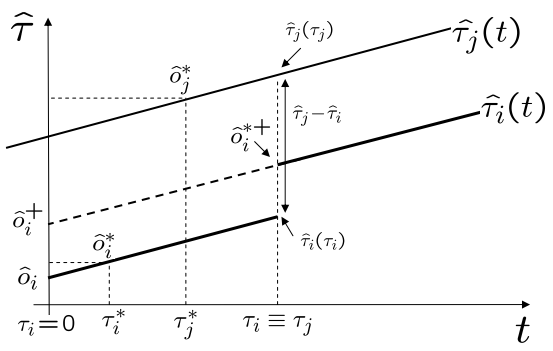

Fig. 5. Offset compensation of node i after receiving synch message from node j at time τ_i graphically represented with an upward rect shift. The variable $\hat{o}_i$ refers to the offset estimate relative to the local time $\tau_i = 0$, Eqn. (16), and the variable $\hat{o}_i^*$ refers to the offset estimate relative to the local time at the last received packet τ_i^*, Eqn. (**??**). Both are equivalent, but the latter is numerically more stable.

C. Offset compensation

According to the previous analysis, after the skew compensation algorithm is applied, the virtual clock estimators have all the same skew, i.e. they run at the same speed. Therefore, the virtual clock estimators of Eqn. (7) can be written as:

$$\hat{\tau}_i(t) = \alpha_v t + \frac{\alpha_v}{\alpha_i}\beta_i + \hat{o}_i \tag{15}$$

At this point it is only necessary to compensate for possible offset errors. Once again, we adopt a consensus algorithm to update the virtual clock offset, previously defined in Eqn. (5), as follows:

$$\begin{aligned} \hat{o}_i^+ &= \hat{o}_i + (1-\rho_o)(\hat{\tau}_j - \hat{\tau}_i) \\ &= \hat{o}_i + (1-\rho_o)(\hat{\alpha}_j\tau_j + \hat{o}_j - \hat{\alpha}_i\tau_i - \hat{o}_i) \end{aligned} \tag{16}$$

where $\hat{\tau}_j$ and $\hat{\tau}_i$ are computed at the same time instant. Fig. 5 graphically illustrates the offset parameter update. According to Eqn. (9), we now need to show that the offset update equation above guarantees that $\lim_{t\to\infty} \hat{o}_i + \frac{\alpha_v}{\alpha_i}\beta_i = \beta_v$ for all nodes. To simplify discussion, let us first define the variable $x_i = \hat{o}_i + \frac{\alpha_v}{\alpha_i}\beta_i$. If we substitute Eqn. (15) into Eqn. (16), we get:

$$x_i^+ = \rho_o x_i + (1-\rho_o)\, x_j \tag{17}$$

which has exactly the same structure of Eqn. (13), therefore under the same hypotheses of the Theorem in the previous section all x_i will converge to the same value, i.e. $\lim_{t\to\infty} x_i = \lim_{t\to\infty} \hat{o}_i + \frac{\alpha_v}{\alpha_i}\beta_i = \beta_v$, where β_v depends on the initial conditions and communication sequence. We can formalize this result in the following theorem:

Theorem 2: Consider the skew update equation given by Equation (16) with arbitrary initial condition $\hat{o}_i(0)$ and $0 < \rho_o < 1$. Also assume there exists $\alpha_v > 0$ such that $\hat{\alpha}_i(t)\alpha_i = \alpha_v$ for all i and t. Also suppose that there exist $T > 0$ such that for any time window of length T each node i in the network transmit at least once to their neighbors their local virtual clock parameters $(\hat{o}_i, \hat{\alpha}_i)$. Finally, assume

that the underlying sensor network communication graph is strongly connected, i.e. there is a communication path from any node i to any other node j in the network. Then

$$\lim_{t\to\infty} \hat{\tau}_i(t) = \hat{\tau}_j(t), \quad \forall(i,j)$$

exponentially fast.
The proof is analogous to Theorem 1 and is therefore omitted.

V. Experimental results

In this section, we present some preliminary results of the Average TimeSync period implemented on Tmote Sky nodes [11] produced by Moteiv Inc. Each node is equipped with the Texas Instrument microprocessor MSP430, an internal oscillator (DCO) running at 1Mhz, and an external oscillator running at 32kHz. In our experiments, we used the local clock provided by the external oscillator, which has a granularity of about $30\mu s$ per tic. As explained above, we used MAC-layer time-stamping to reduce the effects of delivery delay.

In the first set of experiments we used four nodes with a synchronization period $T = 180s$. Every $5s$ an external node simultaneously queried the four synchronizing nodes for their estimated virtual time $\hat{\tau}_i$. In all the experiments we used $\rho_\eta = \rho_\alpha = \rho_o = 0.6$. The results are shown in Fig. 6, where we plotted the error between each virtual clock $\hat{\tau}_i(t)$ from the nodes instantaneous mean $\hat{\tau}_{mean}(t) = \frac{1}{N}\sum_{i=1}^{N}\hat{\tau}_i(t)$. From Fig.6 it is evident the benefit of the skew compensation mechanism. In fact, initially the offset compensation mechanism reduces the offset but the different clock skew show a typical saw-tooth behavior as described in Section I. However, as time passes, all nodes learn their relative skews and use this information to improve performance. Obviously, due to measurement and quantization errors the virtual clocks estimator present some small errors (3-5 tics) which is comparable to the maximum resolution limited by the quantization error (1 tic).

In Figure 7 we plot the values of the virtual clock parameters showing the convergence of the relative skew parameters η_{ij} as well as the skew compensation parameters $\hat{\alpha}_i$, as expected from the theoretical analysis of the ATS algorithm.

As a second experiment we tested the algorithm to cope with dynamic changes of network topology. In particular, four nodes initially separated into two non communicating pairs. As shown in Fig.8, the nodes within the same pair get synchronized, but the pairs drift away. After 30 minutes all the nodes are placed close to each other and they rapidly becomes all synchronized.

The experiments presented here are only preliminary results, and we are currently implementing both protocols on a multi-hop 8x6 sensor network grid. These experiments will be ready by the time of the final submission of the paper.

VI. Conclusions and future work

In this paper we have reviewed the synchronization issues that arise in wireless sensor networks and the current

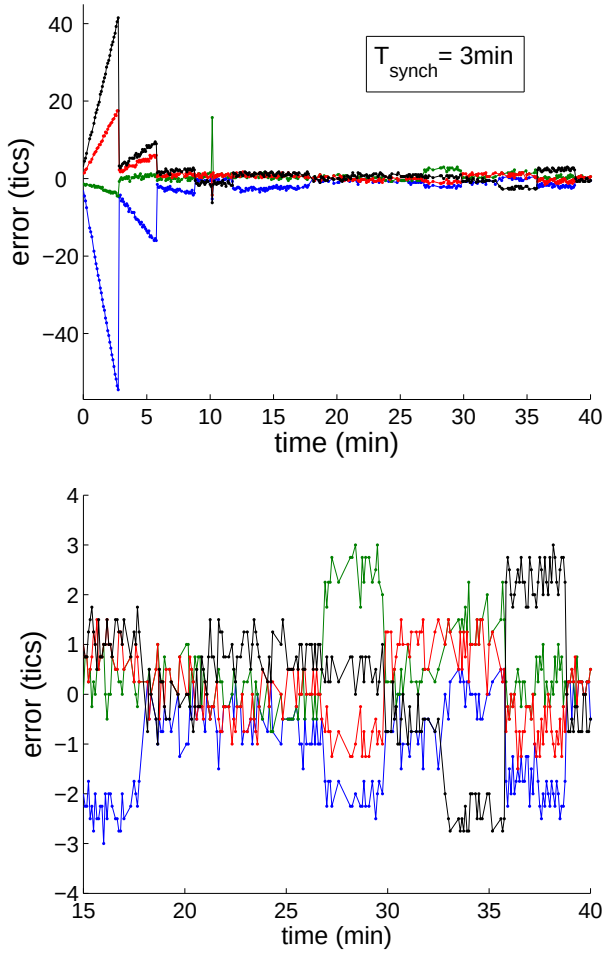

Fig. 6. Error between four nodes from their instantaneous mean $\hat{\tau}_{mean}(t) = \frac{1}{N}\sum_{i=1}^{N}\hat{\tau}_i(t)$ in terms of clock tics (1 tic $\simeq 30\mu s$) *(top)* and zoomed portion during steady state *(bottom)*. Synchronization update period was $T_{sync} = 180s$.

state-of-art in terms of synchronization protocols. We have presented a novel synchronization algorithm, the Average TimeSync protocol, which is based on a very promising class of distributed algorithms known in different research areas as consensus, agreement, gossip or rendezvous algorithm, whose main idea is to average local information to achieve a global agreement on a specific quantity of interest. The proposed algorithm is fully distributed, asynchronous, includes shew compensation and is computationally lite. Moreover, it is robust to dynamic network topology due, for example, to node failure and new node appearance. To our knowledge, only the Distributed Time Synchronization Protocol [15] is fully distributed and provides skew compensation.

However, extensive work is still necessary to compare the performance of our proposed approach relative to FTSP [10] and other protocols over large scale multi-hop sensor network and over longer periods. Moreover, the parameters $\rho_\eta, \rho_\alpha, \rho_o$ have not been optimized to cope with the fact that the clock skews α_i change over time and that there are small measurement time delays. It turns out that these effects correspond

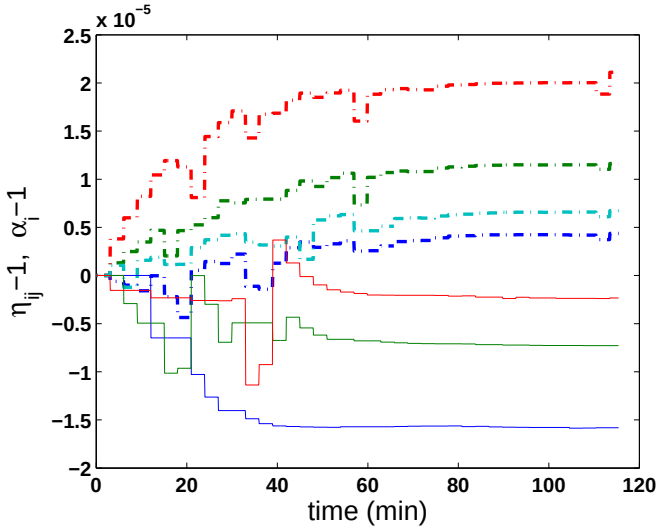

Fig. 7. Virtual clock estimator parameters $\eta_{ij} - 1$ of one node (*thin solid lines*) and $\hat{\alpha}_i - 1$ of the four nodes (*thick dashed lines*) corresponding to experiment on bottom panel of Fig. 6.

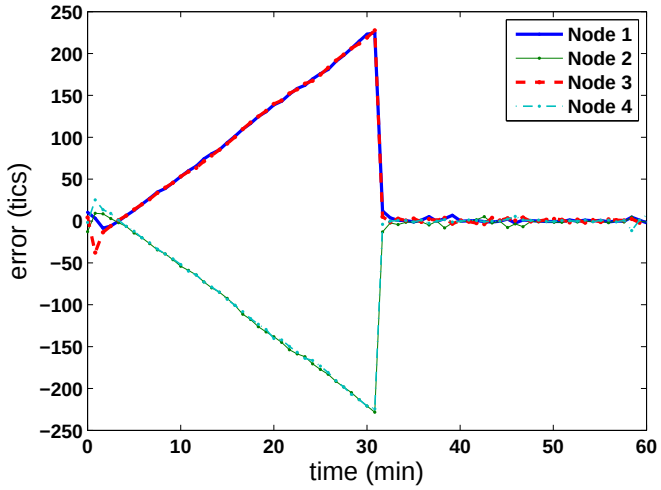

Fig. 8. Error between 4 nodes and their instantaneous mean value for synchronization period $T_{sync} = 30s$. Initially the four nodes are separated in two disconnected pairs: the nodes in each pair get synchronized but the two pairs drifts away from each other. After 30 minutes the two pairs are connected and rapidly all nodes become synchronized.

to add multiplicative noise into the consensus dynamics. We are currently analyzing these effects to compute estimates of the expected synchronization errors as a function of the number of nodes and the communication topology.

ACKNOWLEDGMENTS

The authors would like to thank Alessio Basso for implementing and testing the first version of ATS protocol on the Tmote Sky nodes.

REFERENCES

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory/ACM Transactions on Networking*, 52(6):2508–2530, June 2006.

[2] M. Cao, A. Morse, and B. Anderson. Reaching a consensus in a dynamically changing environment convergence rates, measurement delays and asynchronous events. *submitted to SIAM Journal on Control and Optimization*. [Online] Available at `http://www.eng.yale.edu/controls/pending/flockp2.pdf`.

[3] M. Cao, A. Morse, and B. Anderson. Reaching a consensus in a dynamically changing environment a graphical approach. *submitted to SIAM Journal on Control and Optimization*. [Online] Available at `http://www.eng.yale.edu/controls/pending/flockp1.pdf`.

[4] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI'02)*, pages 147–163, 2002.

[5] S. Ganeriwal, D. Ganesan, M. Hansen, M. B. Srivastava, and D. Estrin. Rate-adaptive time synchronization for long lived sensor networks. In *ACM international conference on Measurement and modeling of computer systems (SIGMETRICS'05)*, 2005.

[6] S. Ganeriwal, D. Ganesan, H. Sim, V. Tsiatsis, and M. B. Srivastava. Estimating clock uncertainty for efficient duty cycling in sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SENSYS'05)*, 2005.

[7] S. Ganeriwal, R. Kumar, and M. Srivastava. Timingsync protocol for sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems (SenSys'03)*, 2003.

[8] A. Giridhar and P. R. Kumar. Distributed clock synchronization over wireless networks: Algorithms and analysis. In *To appear in 45th IEEE Conference on Decision and Control (CDC'06)*, San Diego, December 2006.

[9] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *IEEE and ACM International Symposium on Information Processing in Sensor Networks (IPSN'04)*, April 2004.

[10] M. Maròti, B. Kusy, G. Simon, and Àkos Ldeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, 2004.

[11] Moteiv inc. *Tmote sky data sheet*. [Online] Available at `http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf`.

[12] R. Murray, J. Fax, R. O. Saber, and D. Spanos. Consensus and cooperation in multi-agent networked systems. *to appear in Proceedings of IEEE*, February 2007.

[13] S. Oh, L. Schenato, and S. Sastry. A hierarchical multiple-target tracking algorithm for sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 2197 – 2202, Barcelona, Spain, April 2005.

[14] L. Schenato and S. Zampieri. On the performance of randomized communication topologies for rendezvous control of multiple vehicles. In *Conference on Mathematical Theory of Networks and Systems (MTNS'06)*, Kyoto, Japan, July 2006.

[15] R. Solis, V. Borkar, and P. R. Kumar. A new distributed time synchronization protocol for multihop wireless networks. In *To appear in 45th IEEE Conference on Decision and Control (CDC'06)*, San Diego, December 2006.

[16] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communication of the ACM*, 47(6):34–40, 2004.

[17] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys05)*, pages 51–63, San Diego, CA, USA, November 2005.

[18] J. van Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. In *Proceedings of ACM international conference on Wireless sensor networks and applications (WSNA'03)*, pages 11–19, 2003.

[19] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *ACM Conference on Embedded Networked Sensor Systems (SenSys'05)*, San Diego, November 2005.