Spontaneous Synchronization in Multi-hop Embedded Sensor Networks: Demonstration of a Server-free Approach

Aggelos Bletsas and Andrew Lippman
Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: {aggelos, lip}@media.mit.edu

***Abstract*— A common time reference is essential for distributed sensing, especially when the individual sensors are communicating over wireless, possibly through several intermediate nodes. Prior art in time synchronization over multi-hop wireless sensor networks has focused on centralized solutions that utilize one or more beacons or distributed peer-to-peer approaches, validated only in simulation environments. In this work, we demonstrate distributed synchronization using nearest neighbor communication and no other central point of control. The technique was inspired by natural synchronization in colonies of fireflies and was implemented in an embedded wireless network. The goal was to synchronize speakers and displays at various nodes of the network and quantify the synchronization error as a function of network diameter and communication overhead. One interesting finding was that the error does not scale linearly with the diameter of the network as reported previously in the literature, since it depends on the frequency skew distribution of the participating node oscillators. A video of the demonstration is also available [16].**

I. Introduction

A common time reference is important for many applications of distributed sensing, especially when the individual sensor nodes span a large geographical area and communicate over wireless. Time synchronization becomes non-trivial when individual nodes are several hops away and therefore a single broadcast signal from a particular node (a "server") is not sufficient, as it cannot reach all nodes. Energy constraints of the individual sensor nodes prohibit extensive communication, complicating further the problem of time synchronization. Sensor Networks ought to self-configure and work unattended, therefore any synchronization scheme should have minimal complexity both at the network level (requiring minimal coordination among the nodes) and also at the individual sensor node level, especially due to its embedded, limited computing capabilities microprocessor (as measured in floating point operations per second and internal memory size).

In this work we implement a time synchronization technique for multi-hop, energy/communication/computing-constrained sensor networks which is completely beacon (or server) free. Moreover, it requires no global coordination since all nodes in the network communicate with nearest neighbors for time-synchronization purposes. Therefore the scheme has no centralized point of control (or failure), it has no network routing overhead and it is appropriate for ad-hoc sensor networks where the topology might change (often due to mobility) or might be unknown.

Our initial goal was an experimental evaluation of time synchronization in multi-hop networks, in a real-world setup. For that cause, we implemented a distributed orchestra, where each node could have a speaker to output a song, while at the edges of the network, two nodes were equipped with LED displays (figure 1). At the same time, we wanted to quantify in practice, the observed accuracy and precision of the algorithm against its required communication and computation overhead using our embedded wireless network. Evaluation of the scheme through implementation in a real-world embedded network reveals the important limitations on computation, communication and complexity sensor networks encompass.

On the other hand, evaluations of synchronization schemes only through simulations usually underestimate the limited resources in terms of memory, computation and communication of each node and also assume worst case scenarios that might not reflect reality. Even

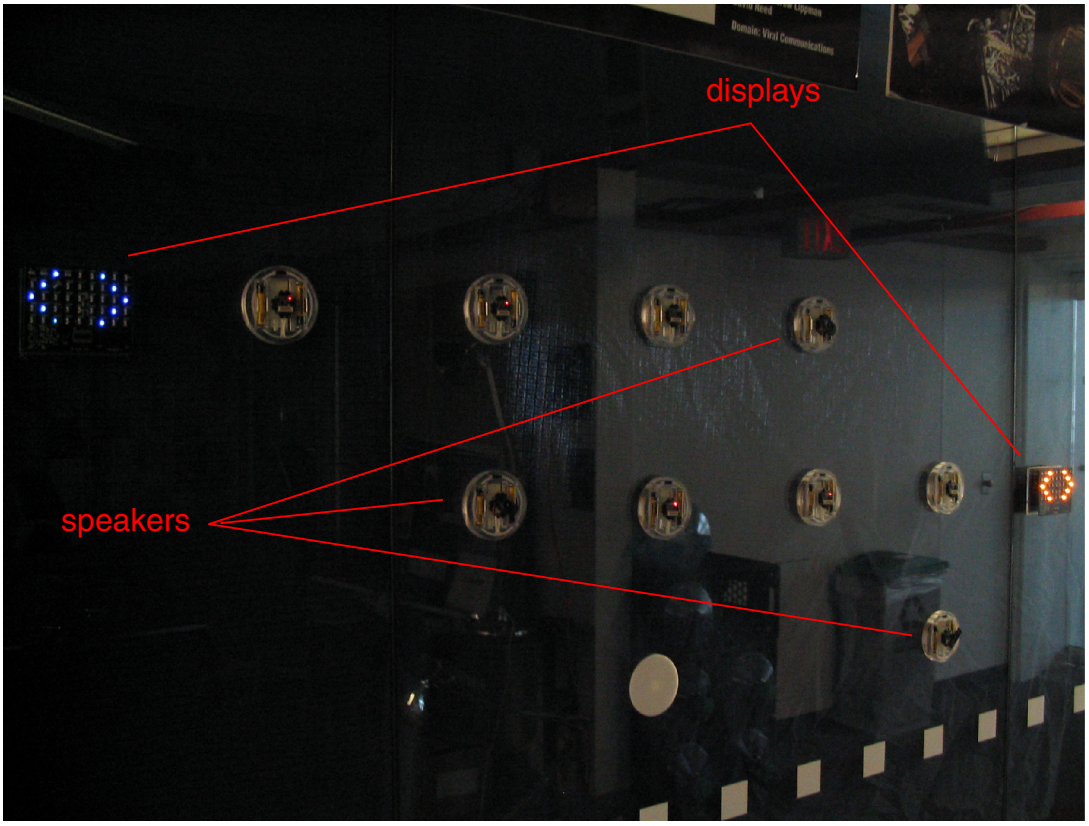
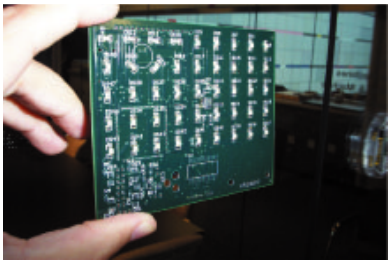


Fig. 1. Demo on a glass wall: each node can communicate with at most 4 immediate neighbors. The network manages to synchronize all nodes so that they can "output" through speakers the same music. At the edges of the network, the nodes are equipped with LED displays instead of speakers, to provide for visual proof of synchrony. All nodes are communicating with immediate neighbors only and there is no point of central control.

(a) 4-IR Pushpin without speaker. The four IR transceivers provide directional communication only along the horizontal and vertical axis.

(b) 4-IR Pushpin with speaker.



(c) 45-LED display. A 4-IR Pushpin is connected behind the LED grid.

Fig. 2. The individual nodes used in this work. Speakers and displays provided for audio-visual output.

though experimental study of time synchronization has been reported before in *single-hop* embedded wireless networks, there is a significant gap in measurements of time synchronization error in realizations of *multi-hop* wireless embedded networks. To our knowledge, this work is the first to fill this gap. Video of the demonstration could be found at [16].

The proposed scheme and the implemented demo were inspired by natural phenomena of synchronization: the way fireflies blink in unison, even though they interact only locally or the way cardiac neurons fire in sync.

II. Requirements and Experimental Setup

The goal in this work was to demonstrate a time synchronization scheme that would be:

a) *transparent* to the sensing or actuating tasks of any node in the network. Each node should communicate only locally with its immediate neighbors and avoid explicit connections to remote servers of "true time" one or more hops away.

b) *self-calibrating* with no coordination requirements upon deployment or during operation. The multi-hop network should *spontaneously* converge to a common time reference without centralized control.

To make matters more realistic, we chose to evaluate the transparent and self-calibrating (as defined above) character of the scheme at the extremes: we evaluated the scheme at the edges of the network, when connectivity is established only through intermediate nodes. RF communication range could be on the order of hundreds of meters, therefore it would be more appropriate to utilize short-range and directive communication links in order to demonstrate multi-hop performance. We used 8051-based micro-controllers (8-bit, 2 Kbytes of RAM and 32 Kbytes of program space) connected to short-range, 4-way infrared transceivers. Those are the pushpin nodes [11], [18], that we packaged in round battery holders as shown in figure 2(a). Pushpins practically allowed evaluation of the synchronization scheme at the edges of the network, for several values of network diameter d as shown in figure 3(a). The experimental setup for $d = 4$ is shown in 3(b).

The goal was to demonstrate network multi-bit clock synchronization among all nodes in a distributed fashion, not just synchronization to a reference signal coming from a specialized server [12], [1] or beacon [4]. No prior knowledge of network topology was assumed and all nodes were loaded with the same code. All nodes could be equipped with small speakers (figure 2(b)) and as a proof of synchrony they would play the same piece of music at the same time. According to ([3] p.95), the smallest perceivable time difference from humans is on the order of 30-50 milliseconds, therefore clock synchronization error above that limit could be perceived.

Apart from the oscilloscope measurements at the edges of the network and the audio outputs at many intermediate nodes, visual patterns at the edge nodes provided for visual proof of synchrony. Displays from the rf-Badges [9], [17] were connected to 4-IR pushpins and used in this work (figure 2(c)).

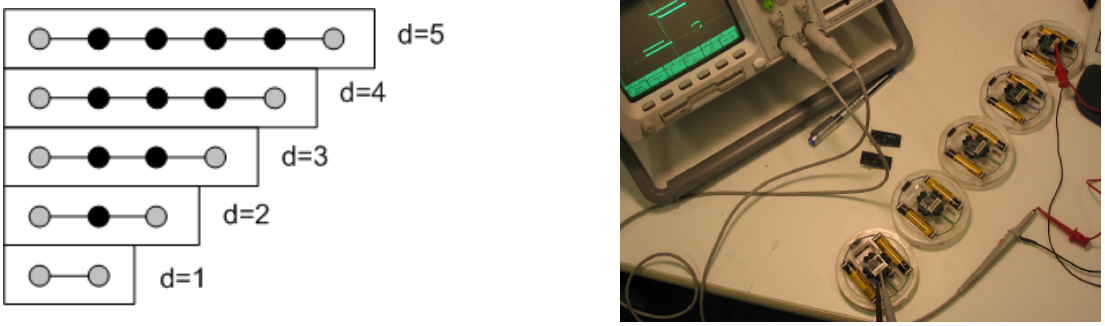
III. The Algorithm and its Implementation in our Embedded Network

Lamport in his 1978 work in the context of computer clocks and processes synchronization [10], described a simple algorithm, based on the fact that *time* is a strictly monotonically-increasing quantity. Therefore events happening in subsequent times should have timestamps ordered accordingly, otherwise a correction in the clocks should be made. Although Lamport's work has been extensively referenced in the area of *sensor network time synchronization*, there has been no validation and testing in embedded networks so far (at least to the extent of our knowledge). Since time is viewed as a non-decreasing quantity in Lamport's algorithm, its implementation probably has been considered problematic in memory-restricted and communication-constrained sensor networks.

- **Broadcast:** node i transmits its clock value $C_i(t)$ at regular intervals. Time-stamping occurs just before transmission and the MAC protocol has been modified accordingly.
- **Receive and Compare:** upon reception from node j of a clock value $C_i(t)$ from node i, node j compares and keeps the highest value: if $C_i(t) > C_j(t)$ then $C_j(t) \leftarrow C_i(t)$ else ignore.

In this work, we modify Lamport's algorithm to fit the memory and communication constraints of sensor networks and through implementation in a multi-hop, embedded network, we prove that the new algorithm can sufficiently synchronize the whole network, in a distributed, *transparent* and *self-calibrating* way (as defined previously), satisfying many real-world scenarios.

The first modification in Lamport's algorithm is that time is no longer considered a monotonically increasing quantity: clock $C_j(t)$ in every network node j is bounded above and upon reaching that value, time is



(a) (b)

Fig. 3. Topologies for various network diameters d used in this work 3(a). The oscilloscope probes are connected at the edge nodes of the network. The case for $d = 4$ is shown in 3(b).

reset. Therefore, clock function $C_j(t)$ follows a "saw-type" periodic waveform and its period should be set according to the natural phenomenon which is sensed by the sensor network. In this work, since the goal was distributed synchronized play of music, the period T of each clock was set to 13 seconds approximately.

The first reason behind upper bounding time, was the fact that timestamps are communicated among neighboring nodes and therefore their size in bits should be kept minimal, because of memory, bandwidth and energy constraints. In this work, clock value $C_j(t)$ of node j is represented by an unsigned 16-bit variable, incremented each time a 16-bit *counter* resets. This reset occurs every 5.9 msecs approximately, limiting the resolution of each clock variable $C_j(t)$ in the millisecond regime. The counter is interrupt driven and since it controls time increments, it is assigned the highest priority interrupt.

The second reason behind upper bounding time, was our desire to explicitly study *self-calibration* capabilities of the algorithm and show in practice that even though clocks reset periodically (in this case, every 13 seconds), the network as a whole, re-synchronizes quickly and unattended (*spontaneously*) and is able to perform its sensing and actuating tasks.

Note that in this realization, we have time $C_j(t)$ of node j to be represented as a 16-bit integer, with resolution set by another 16-bit counter. However, only the first 16-bit value is communicated to nearest neighbors. The length in bits of the clock value $C_j(t)$ and its resolution depend on the physical phenomenon that needs to be sensed. For example, for environmental sensing of moisture, a 16-bit clock incremented every 1.3 seconds would need 24 hours approximately, to reset. Therefore, the same "saw-type" definition of time would suffice, the information communicated over the network would be the same as in the example of this paper and the only modification would be in the clock resolution in each node. The network would reset in synchrony every 24 hours instead of 13 seconds. The slower period in this work (and resolution on the order of milliseconds) helped us quickly validate the fact that the network *re-calibrates* after every clock variable $C_j(t)$ expiration, without unwanted periods of instability.

In other words, the length in bits and the resolution of the clock variable $C_j(t)$ depend on the physical phenomenon to be sensed and the algorithm could be used with success in many different contexts and applications such as environmental sensing.

The second modification in Lamport's algorithm, is the fact that broadcasting of time-stamped information is controlled by an independent timer and not by the clock of each node. The reason behind such implementation decision was that we wanted to decouple the two stages of the algorithm (broadcast and receive), simplify design and avoid *bootstrapping* problems, that might occur if we had used the same timer to control both *when* as well as *what* to transmit. Time-stamping during the broadcast phase occurred just before transmission, therefore the Medium Access Control (MAC) protocol in every node had been modified accordingly.

Table I lists the clock period T and the resolution of each node's clock, the time needed for each node to transmit timing packet information to its neighbors and how often every node broadcasts its clock value in packets per second (pps), for two scenarios ($r1,\ r2$) evaluated in this work.

TABLE I

PERIOD AND RESOLUTION OF EACH CLOCK, TRANSMISSION DELAY AND BANDWIDTH USED FOR TIMING PACKETS (IN PACKETS PER SECOND).

	T of $C(t)$	res of $C(t)$	tx delay	bw
r1	13.2 sec	5.9 msec	1.24 msec	0.3 pps
r2	13.2 sec	5.9 msec	1.24 msec	3 pps

It is important to note that the packet each node transmits at regular intervals $\frac{1}{bw}$, contains only the 16-bit time variable, a protocol header byte and one additional byte with Cyclic Redundancy Check (CRC) information. In other words, the 4-byte packet transmitted contains no information about node source id, destination id or any other kind of routing information since communication is happening with nearest neighbors. Therefore, the synchronization scheme is *transparent* (as defined previously) to the sensing or actuating task of each node.

We called the new scheme "Spontaneous Synchronization". Video of the demonstration could be found at [16].

IV. RESULTS

We run experiments with duration 500 seconds each and measured the absolute synchronization error $\mid C_i(t) - C_j(t) \mid$ where nodes i, j are the edge nodes of the network as shown in figures 3(a),3(b). To do so, each node output a pulse when its clock variable reached a specific value ($C_i(t) = max/2$)[1]. We have already described that time is represented by an unsigned 16-bit integer (reaching its maximum value and then reseting every T seconds), incremented from the overflow of a 16-bit counter (controlled by the crystal oscillator of each node and overflowing every res milliseconds, from table I). Therefore, we measured the absolute synchronization error at the edges of the network every T seconds and for $T \simeq 13\ sec$, the 500 seconds experiment corresponded to 37 measurements per experiment.

The network managed to synchronize all individual nodes so they could play the same piece of music repeatedly, as long as the nodes were switched on. That provided a quick proof of synchronization error smaller than 30 milliseconds, since that is the smallest time difference perceived by humans ([3], p.95). Moreover, we were assured that time reseting at each individual node didn't cause instabilities but on the contrary, the network managed to re-calibrate and converge to a common time reference, continuously and unattended.

[1]note that max need not be $2^{16} - 1 = 65535$ but it could be set to a smaller value: $max = \frac{T}{res}$.

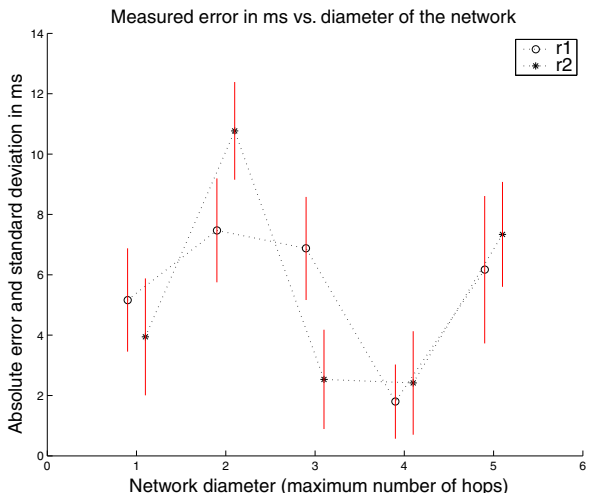


Fig. 4. Measured average time synchronization absolute error and its standard deviation in milliseconds, as a function of network diameter. Clock resolution and transmit time is on the order of milliseconds, limiting the error in the millisecond regime, as expected. Notice that error is not increased linearly with number of hops, since error depends on the sign of clock skew differences between neighboring nodes (second term in equation 5).

The oscilloscope measurements helped us quantify the performance of the synchronization scheme (figure 3). Average absolute error $\widehat{|\epsilon(t)|}$ and its standard deviation for different network diameters are shown in figure 4. All experiments were run twice since apart from network diameter (d), we wanted to study performance against different bandwidth (bw from table I) used for broadcasting time (broadcast phase of the algorithm).

From figure 4 we can see that the absolute synchronization error $|\epsilon|$ is on the order of a few milliseconds. This is not a surprising result since the clock resolution of each network node is on the order of milliseconds (table I). Moreover, as we will see below, the synchronization error depends on the transmission delay which is, again, on the order of milliseconds. Ways to reduce the error because of those two factors down to the μsecond regime are discussed in section V.

What is surprising about these measurement results, is the fact that synchronization error does NOT increase linearly with the diameter of the network as it has been reported previously in simulation setups. A simple analysis follow to justify the above findings: we could model the timer $C_i(t)$ of each node i as a linear function. Time increases with a rate ϕ_i that depends on the crystal

oscillator of each node. The difference $\phi - 1$ is called frequency skew and for the crystals used in our nodes, it is on the order of $\pm$ 50 parts per million (ppm).

Let's ignore for now the fact that time resets at each node and let's assume that node i transmits its timestamp at time t_0. The packet will be received and processed by neighboring node j at time $t_0 + x$.

Time duration x includes the propagation time of the signal which is basically the time needed for the first bit to arrive at the destination (distance/speed of light), the transmission time which is the time needed for the transmitter electronics to transmit the waveform (tx delay at table I in section III) and finally the time the operating system at the receiver needs to process the received packet.

$$C_i(t_0) = \phi_i \; t_0 + \theta_i \quad (1)$$

$$C_j(t_0 + x) = \phi_j \; (t_0 + x) + \theta_j \quad (2)$$

$$x = propagation\ delay\ + \quad (3)$$
$$+\ transmission\ delay\ +$$
$$+\ operating\ system\ delay$$

Propagation delay is negligible, on the order of a couple of μseconds for short range transceivers, therefore x is dominated by tx delay and os delay. In our system, tx delay is 1.24 msec (since we are using slow transceivers) while operating system delay has been kept one order of magnitude smaller, given the fact that we are using pipelined, RISC micro-controllers driven by 22.11084 MHz crystals. Medium Access Control has been modified in order to avoid adding delays in the transmission of timing packets.

If $C_i(t_0) > C_j(t_0 + x)$ then $C_j(t_0 + x) \leftarrow C_i(t_0)$ and the absolute error $\mid \epsilon \mid$ at time $t_0 + x$ becomes:

$$\begin{aligned} \mid \epsilon(t_0 + x) \mid &= \mid C_i(t_0 + x) - C_j(t_0 + x) \mid \\ &= \mid C_i(t_0 + x) - C_i(t_0) \mid \quad \Rightarrow \\ \mid \epsilon(t_0 + x) \mid &= \phi_i \; x \end{aligned} \quad (4)$$

Therefore, the error at time $t_0 + x$ is on the order of $(1 \pm 50 \; 10^{-6}) \; x \approx tx \; delay = 1.24 \; msec$. Thereinafter, the error might increase or decrease depending on the frequency skew differences of node i, j clocks, since it is not difficult to see that according to this linear representation of time in equation 1, the error at time $t_c > t_0$ becomes[2]:

$$\begin{aligned} \epsilon(t_c) &= C_i(t_c) - C_j(t_c) = \\ &= \epsilon(t_0 + x) + (\phi_i - \phi_j) \; \Delta t \end{aligned} \quad (5)$$

$$\Delta t = t_c - (t_0 + x) \quad (6)$$

We can see that the error at time t_c might decrease if $\phi_i - \phi_j < 0$ or increase if $\phi_i - \phi_j > 0$. The amount of increase or decrease is on the order of $(50 \; 10^{-6} - (-50 \; 10^{-6})) \; T/2 \approx 650 \; \mu sec$ since we have at least one packet transmission per T seconds. From the above, it is straightforward to understand that the measured absolute error might decrease below *tx time* and there were occasions when the absolute error could drop at the $\mu second$ regime.

The fact that time resets at each node doesn't affect the above analysis: reseting changes θ at each clock, not ϕ (which depends on the crystal oscillator on-board) and time differences using our algorithm depend on frequency skew differences $\Delta\phi$ (equation (5)), therefore changes of θ due to reseting, don't matter.

Even in the case where a node's clock resets and then that node receives a clock value from another node's clock which is close to reset, it can be seen that there are no instabilities in the overall system since both clocks will eventually reset and the synchronization error between them will start from $\phi \; x$ and will be increased or decreased depending on the sign of their frequency skew difference.

From figure 4 we can see that increasing the broadcasting rate from 0.3 packets per second (r1) to 3 packets per second (r2), doesn't dramatically affect the overall error, since that increase of rate just decreases Δt in equation (5) but it doesn't affect x which is the dominating factor in the error. Increasing the broadcast rate (or decreasing Δt) allows for finer increase or decrease of the error (on the order of 650 μsec/10 = 65 μsec for r2 compared to 650 μsec for r1). Increasing the broadcast rate would make more sense for oscillators with higher frequency skew, than those used in this work ($\pm$ 50 ppm).

From the above analysis, it is now obvious why the average absolute error is not increasing monotonically with the diameter of the network. That is because the error as we saw, depends on the sign of the frequency skew among the clocks (second term in equation 5), therefore by inserting additional nodes in a chain topology (figure 3(a)), the sign might be negative, leading to

[2]provided that there is no time modification during the receive-and-compare phase of the algorithm at node j

smaller synchronization errors. Analysis that shows that error increases linearly with the diameter of the network [10] assumes worst case scenarios i.e. the sign of $\Delta\phi$ in equation (5) is always positive, therefore the error builds up with the number of hops. This interesting behavior as depicted in figure 4 would not have been observed if we hadn't implemented our algorithm in a real-world embedded network.

V. Further Improvements

The synchronization error could be further reduced by minimizing x. That can be achieved if the packet transmission time (which is deterministic and known) is incorporated in the transmitted timestamp during the broadcast phase of the algorithm. That basically means that each node broadcasts at time t, $C(t) + tx\ time$ instead of $C(t)$. Moreover, the operating system delays could be minimized or anticipated (and therefore incorporated as well in the transmitted timestamp). It is also useful to reduce uncertainties due to the channel access scheme in the MAC layer (allowing for time-stamping at the MAC layer could be one solution).

We implemented the above modifications in a RF, embedded, single-hop network and the synchronization error was reduced down to the μsecond regime. The interested reader could refer to [2] for additional information regarding the RF, single-hop case.

VI. Spontaneous Order and its Connection to Biological Synchronization

What we have seen so far, is that coupling between neighboring oscillators with similar (but not exactly the same) frequency skew and periodic (due to reset) time waveforms, is able to globally provide network synchrony.

This global phenomenon of sync emerged as a consequence of local interactions between homogeneous elements and resembles similar phenomena found in nature: the way fireflies manage to globally blink in unison, even though they interact locally or the way millions of cardiac neurons fire in sync to produce the cardiac pulse. Those phenomena depend on coupling between oscillators, they have nothing to do with averaging of similar quantities (like timestamps for example) and they are canonical examples of *entrainment* ([15], p.72). In the above examples of entrainment, including our work, synchrony is not controlled by any centralized authority but it is the natural emergent result of local interactions.

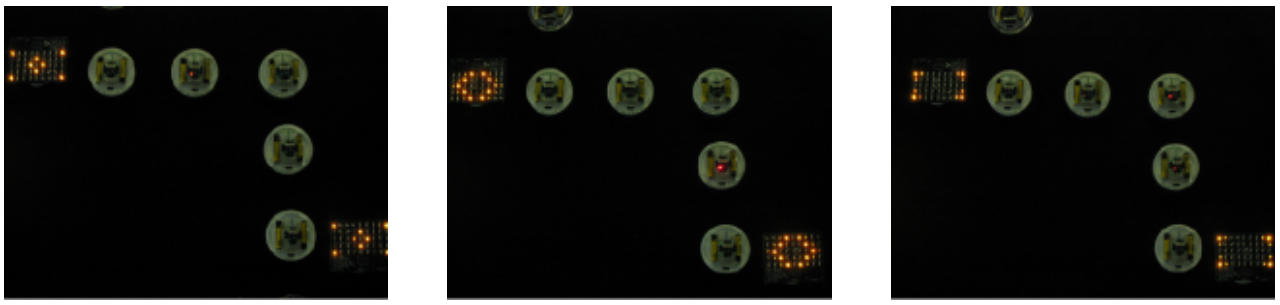
Inspired by the fireflies phenomenon, we attached two display-equipped nodes (figure 2(c)) at the edges of the network in order to visualize synchrony (figure V). The displays output a "heartbeat waveform" synchronized by the distributed scheme presented in this work. The difference with fireflies is that fireflies need only 1-bit synchronization as opposed to the 16-bit synchronization presented in this work.

VII. Relevant Work and Discussion

We were particularly interested in multi-hop time synchronization algorithms. Work using simulations and reported in [14], [6] and [7] falls into this category. The basic idea is that each node exchanges two-way timing information with its closest neighbors, its neighbors with their neighbors and so on, up until the reference node. In [14], the same clock model of equation 1 is used, time offset between two nodes is linear (as we also saw in equation 5) and bounds on the time offset between two nodes are derived. Those bounds are used to outcast redundant information and estimate in a sub-optimal, low complexity scheme the offset between any two nodes. In [6], the same clock model is used and two-way timing information is used to estimate time offset between two neighboring nodes using the NTP algorithm [12]. The major problem with such approach is the fact that delays between any two nodes are not symmetric in general. In [7], two versions of pair-wise synchronization are used: one based on the implementation of a spanning tree starting from the reference node down to the edges and one distributed implementation using node-to-node measurements from a node that needs synchronization up to the reference node.

In all the above implementations there are two distinct differences compared to our Spontaneous approach: a) the above need two-way measurements, meaning that each node needs to send a timing packet and receive its response back, while in our case we need just one way transmissions and therefore, the overhead is smaller, b) all the above approaches need the maintenance of a hierarchy from the edge nodes up to the reference node in the form of a Network Time Protocol(NTP)-like hierarchy ([6], [14] or Spanning tree version in [7]) or find appropriate communication paths toward the reference path (distributed version in [7]). This is significant overhead compared to our approach where routing is not needed, since the edges of the network are coupled through nearest neighbor communication and no relay of timing packets is required.

In [13], it was suggested that sensor network nodes need to provide information about when a specific event happened, according to their own clock instead of trying



(a) (b) (c)

Fig. 5. Visual proof of synchrony. A "heartbeat" pattern is synchronized over the network and displayed at the edges. The distributed, server-free approach for network synchronization resembles the decentralized coordination of colonies of fireflies that inspired this work.

to synchronize all nodes to the same reference. Ordering of events matters, according to the same reasoning and each node should be able to provide an interval, according to each own clock, on when a specific event happened as an approximation of global time. Bounds provided by that scheme are a function of network diameter and specific formulas were derived. However, approaches like that make distributed actuation scenarios (like those in our work) difficult to implement in practice.

Reference signals could be used to trigger time-stamping in different receivers, reducing the variability of a wireless transmission time due to channel access delay and propagation time of a signal [4]. Then the two receiving nodes would exchange information to find out their time offset. Such approach is quadratic in the number of participating nodes and therefore more expensive than the approaches discussed so far. It also needs special handling in the case of several beacons and nodes in the vicinity of more than one beacon. Also it is not an infrastructure-free approach compared with our Spontaneous scheme.

It has been argued that time synchronization might be viewed as an offline problem, depending on the application [5]. *Post-facto* synchronization in [5] basically suggests that clocks should be left "free running" and time offsets should be calculated offline, after data have been time-stamped and gathered. This is true for a large range of potential wireless sensor network applications.

Finally, we should mention the cooperative construction of a propagating signal waveform across several hops, that could be used for synchronization [8]. It remains to be implemented and evaluated in practice.

VIII. Conclusion

We presented and practically implemented a simple time synchronization algorithm for wireless mutli-hop sensor networks, which requires no beacons/servers in the network and no global coordination but only local communication among the nodes. We quantified its error as a function of the diameter of the network and theoretically analyzed the observations. Within certain limitations, the algorithm could practically provide for 16-bit synchronization in realistic embedded sensor networks, with msec down to μsec synchronization error without extensive communication or computing overhead requirements. We experimentally showed that the synchronization error is not necessarily increased linearly with the number of hops and provided a simple analytical explanation. Video of the demonstration could be found at [16].

Acknowledgment

The authors would like to thank Josh Lifton and Mat Laibowitz from the Responsive Environments Group (http://www.media.mit.edu/resenv/) for their kind assistance in hardware throughout this work.

References

[1] A. Bletsas, *"Evaluation of Kalman Filtering for Network Time Keeping"*, Proceedings of IEEE International Conference on Pervasive Computing and Communications (PERCOM), Fort-Worth Texas, March 2003.

[2] A. Bletsas, A. Lippman, *"Natural Spontaneous Order in Wireless Sensor Networks: Time Synchronization Based on Entrainment"*, Technical Report, MIT Media Lab, December 2003.

[3] P. R. Cook, *Music, Cognition and Computerized Sound. An Introduction to Psychoacoustics*, MIT Press, Cambridge Massachusetts, 1999

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

[5] J. Elson, K. R$\ddot{o}mer$, *"Wireless Sensor Netowkrs: A New Regime for Time Synchronization"*, Proceedings of the First Workshop on Hot Topics in Networks, October 2002.

[6] S. Ganeriwal, R. Kumar and M. Srivastava, *"Timing Sync Protocol for Sensor Networks"*, Proceedings of the First International Conference on Embedded Networked Sensor Systems, Los Angeles CA, 2003.

[7] J. V. Greunen, J. Rabaey, *"Lightweight Time Synchronization for Sensor Networks"*, Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications, San Diego, CA, September 2003.

[8] A. Hu, S. D. Servetto, *"Asymptotically Optimal Time Synchronization in Dense Sensor Networks"*, Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications, San Diego, CA, September 2003.

[9] M. Laibowitz, J. Paradiso, *Wearable Wireless Transceivers*, Circuit Cellar, no.163, February 2004.

[10] L. Lamport, *Time, Clocks, and the Ordering of Events in a Distributed System*, Communications of the ACM, vol. 21, no.7, July 1978.

[11] J. Lifton, D. Seetharam, M. Broxton, J. Paradiso, *"Pushpin Computing System Overview: a Platform for Distributed, Embedded, Ubiquitous Sensor Networks"*, Pervasive 2002, Proceedings of the Pervasive Computing Conference, Zurich Switzerland, August 2002.

[12] D. L. Mills, *"Network Time Protocol (Version 3) Specification, Implementation and Analysis"*, RFC 1305, University of Delaware, March 1992.

[13] K. R$\ddot{o}mer$, *"Time Synchronization in Ad Hoc Networks"*, MobiHOC 2001, Long Beach Ca.

[14] M.L. Sichitiu, C. Veerarittiphan, *"Simple, Accurate Time Synchronization for Wireless Sensor Networks"*, IEEE WCNC 2003.

[15] S. Strogatz, *Sync, The Emerging Science of Spontaneous Order*, 1st ed. New York: Theia, 2003.

[16] Demo website, http://web.media.mit.edu/ aggelos/demos/demos.html

[17] Badge website, http://www.media.mit.edu/resenv/badge

[18] Pushpin website, http://web.media.mit.edu/ lifton/PushPin/