# Efficient Consensus Synchronization via Implicit Acknowledgment

Andrew G. Klein
Department of Engineering and Design
Western Washington University
Bellingham, WA 98225
Email: andy.klein@wwu.edu

D. Richard Brown III
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: drb@wpi.edu

***Abstract*—A technique for achieving synchronization in wireless networks using only existing traffic is developed. Prior work has either ignored propagation delay, or has required bidirectional messages consisting of explicitly acknowledged unicast transmissions. We develop an approach using "implicit acknowledgment" that achieves precise consensus synchronization by exploiting the broadcast nature of the wireless medium. This significantly reduces the number of transmissions needed for synchronization throughout the network, and is applicable to networks with unacknowledged multicast and broadcast traffic. Results suggest the technique is effective for precise, low-overhead network synchronization, and numerical results are presented for two particular network configurations.**

## I. Introduction

Synchronization in wireless networks is necessary to enable coordination among the nodes, thereby facilitating scheduling of communication resources, interference avoidance, event detection/ordering, data fusion, and coordinated wake/sleep cycles [1]. Most modern wireless communication standards, e.g. 802.11, 802.15, and 802.16, use network synchronization for various functions including time-slotting and power management. The synchronization process in a network is almost always viewed as a separate process from the normal operation of the network and is typically achieved through a dedicated synchronization protocol. A classic example is Network Time Protocol (NTP) [2] which requires the establishment of a hierarchical structure and the periodic exchange of dedicated synchronization messages between devices at different layers in the hierarchy. Over the last 30 years, a variety of dedicated synchronization protocols have been developed including NTP, Precision Time Protocol (PTP) [3], the Global Positioning System (GPS) [4], and several lightweight protocols for sensor networks, e.g. [1], [5], [6]. These dedicated synchronization protocols achieve various tradeoffs between overhead, complexity, and accuracy.

In the absence of a dedicated synchronization protocol, nodes can still learn their *relative* clock offsets via existing network traffic. For example, the class of timestamp-free synchronization techniques which were originally studied in the context of natural phenomena, e.g. synchronization of the firing rates of fireflies, led to formal mathematical models for systems of pulse-coupled oscillators in [7] and application of these models to wireless sensor networks in [8], [9]. While these studies represented an exciting paradigm shift with respect to the prior work, a limitation of the pulse-coupled oscillator literature is that it is based on unidirectional transmissions and therefore assumes negligible propagation delays. This inherently limits the synchronization accuracy of these methods.

To account for propagation delay, relative clock offsets can also be gleaned from timestamps in packet traffic or from the physical layer characteristics of the transmissions. Recently, a synchronization approach was proposed that operates via a series of pairwise message exchanges that cause clock drifts and offsets between pairs of nodes to converge toward a consensus clock at an exponential rate, without requiring a dedicated synchronization protocol [10]. However, the approach relies on the assumption that all network traffic consists of unicast transmissions that are explicitly acknowledged.

In this paper we propose an enhanced synchronization scheme which accounts for propagation delay, builds upon the work in [10], does not require explicit acknowledgments (ACKs), and exploits the broadcast nature of the wireless medium to roughly halve the number of transmissions required to reach a consensus clock. Such a synchronization scheme is particularly suitable in networks which do not employ ACKs, for example in networks with multicast and broadcast traffic or in networks without guaranteed delivery (e.g. when connectionless transport layer protocols, such as UDP, are employed).

With the absence of a dedicated synchronization protocol, we use a probabilistic Markov model of network traffic that reflects the random nature of transmissions. Our results suggest that non-hierarchical embedded synchronization techniques can be effective for low-overhead network synchronization. Numerical examples showing convergence and divergence of consensus synchronization in two different network configurations are also provided.

Notation: Vectors and matrices are denoted by boldface letters. $\boldsymbol{I}_N$ denotes the $N \times N$ identity matrix, $\|\cdot\|$ represents the Euclidean norm of the enclosed vector, and we use $(\cdot)^\top$ for transposition.

This work was supported by the National Science Foundation awards CCF-1302104 and CCF-1319458.

## II. System Model

We assume a time-division duplexed (TDD) network of $N$ nodes and denote the propagation delay from node $i$ to node $j$ as $\psi_{i,j}$. Since all of the channels in the system are TDD, we assume reciprocal propagation delays $\psi_{i,j} = \psi_{j,i}$ in each link. Basic electromagnetic principles have long established that channel reciprocity holds at the antennas when the channel is accessed at the same frequency in both directions [11].

### A. Random Asymmetric Gossip Model

With no dedicated synchronization protocol, we assume the nodes take turns transmitting in a probabilistic order described by an irreducible Markov chain. By letting $s_k \in \{1, \ldots, N\}$ denote the random index of the node transmitting in the $k$th active TDD time slot, we specify a stochastic matrix $\boldsymbol{P}$ with $i, j^{\text{th}}$ entry $p_{i,j}$ corresponding to the probability that $s_k = j$ given that $s_{k-1} = i$. While in practice a node may transmit twice in succession, such repeated transmissions are not useful to our proposed synchronization protocol and are effectively nuisance transmissions; thus, for notational simplicity we assume that $p_{i,i} = 0$ for all $i$. Since $\boldsymbol{P}$ is a stochastic matrix, $\sum_j p_{i,j} = 1$ for all $i$. In the case where all nodes have equal probability of transmission, $\boldsymbol{P}$ consists of zeros along the diagonal, and all off-diagonal entries equal to $1/(N-1)$. This framework is flexible enough to accommodate a wide range of network types, including deterministic schedules of transmission. For example, a round-robin schedule results if $\boldsymbol{P}$ is chosen to be a circulant shift of the identity matrix. This framework can be represented as a random walk on a weighted directed graph $\mathcal{G}$ with vertices $V = \{1, \ldots, N\}$, edges $E \subseteq V \times V$, edge weight $p_{i,j}$ on edge $(j, i)$, and adjacency matrix $\boldsymbol{P}^\top$. Finally, we do not require that the graph is completely connected, but we do require that the Markov chain is irreducible, resulting in a strongly connected graph. In addition, if *either* $p_{i,j} \neq 0$ or $p_{j,i} \neq 0$, we assume nodes $i$ and $j$ can reliably *receive* transmissions from one another.

### B. Reference Time and Local Time

The nodes in the network do not possess a common notion of time. We use the notation $t$ to refer to some notion of reference time, i.e. the "true" time, in the system. All time-based quantities such as propagation delays and/or frequencies are specified in reference time unless otherwise noted.

None of the nodes have knowledge of the reference time $t$. The local time at node $i$ is modeled as $t_i = t + \Delta_i(t)$ where $\Delta_i(t)$ is a non-stationary random process that captures the effect of clock drift, fixed local time offset, local oscillator phase noise, and frequency instability [12]. Note that the goal of consensus synchronization is not to synchronize each node such that $t_i \to t$. Rather, the goal is to synthesize a consensus clock $\bar{t}$, which is itself a function of $\{t_1, \ldots, t_N\}$, and to synchronize each node such that $t_i \to \bar{t}$.

Over short time periods, a reasonable first-order model of local time can be written as $t_i = \beta_i t + \Delta_i$ where $\beta_i$ represents the nominal relative rate of the clock at node $i$ with respect to the reference time and $\Delta_i$ is the local clock offset at $t = 0$. None of the nodes have knowledge of $\beta_i$ or $\Delta_i$.

## III. Network Synchronization

This section describes a network synchronization protocol that allows each node in the network to arrive at a common clock drift $\beta_i$ and clock offset $\Delta_i$ through random pairwise message exchanges. The goal is not to force $\beta_i = 1$ and $\Delta_i = 0$ or to achieve "average consensus" where the global average of the drifts and offsets are preserved over time [13]. Rather, the goal is to drive the clock drifts and offsets to *common* values $\bar{\beta}$ and $\bar{\Delta}$ across the network. For conceptual simplicity, we describe the network synchronization protocol as a two-step process: (i) drift compensation and (ii) offset compensation. In practice, both drift and offset compensation can be performed simultaneously, since pairwise drift estimates can be inferred "for free" from physical layer characteristics of normal network traffic.

We do not restrict ourselves to a particular method for drift and offset estimation, and we simply assume that when a random node $s_k$ transmits in time slot $k$, the node $s_{k-1}$ that transmitted in the previous time slot adjusts its local clock based on its local estimate of its pairwise drift and offset with respect to node $s_k$. Due to the broadcast nature of the wireless medium, each transmission from node $s_k$ at time $k$ effectively serves three purposes: (i) provides inherent feedback in the form of an implicit acknowledgment which node $s_{k-1}$ uses to adjust its clock, (ii) blindly initiates the next implicit synchronization exchange with whichever node $s_{k+1}$ transmits next, and (iii) transmits regular "intended" traffic to one or more other nodes in the system. The protocol requires that nodes $s_{k-1}$ and $s_{k+1}$ are able to reliably receive transmissions from node $s_k$, which is guaranteed by the assumptions in the probabilistic messaging framework of Section II-A. Even though the transmission from node $s_k$ at time $k$ implicitly provides information to nodes $s_{k-1}$ and $s_{k+1}$ as part of the synchronization approach, we reiterate that node $s_k$'s broadcast transmission may well contain regular traffic intended for yet another node or nodes besides nodes $s_{k-1}$ and $s_{k+1}$. An example of this sequence of exchanges is shown in Fig. 1, including the implicit "overheard" message exchanges used for synchronization, as well as the regular "intended" network traffic.

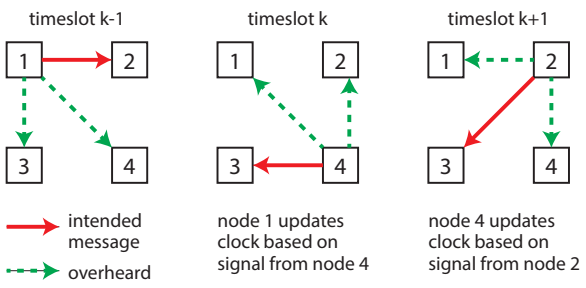


Fig. 1. Implicit message exchanges between nodes 1 and 4, and then nodes 4 and 2.

### A. Step 1: Drift Compensation

Since nodes derive their symbol rate and carrier frequency from the same local oscillator that drives the local clock, any message between a pair of nodes in the network allows for the estimation of pairwise clock drift at the physical layer through carrier frequency or symbol rate offset estimation. Pairwise clock drift can also be estimated at the MAC layer through observing multiple timestamped messages from another node in the network.

We define the drift vector at time $k$ as

$$\boldsymbol{\beta}[k] := [\beta_1[k], \ldots, \beta_N[k]]^\top \in \mathbb{R}^N. \tag{1}$$

While none of the nodes know their local drift or the drift of other nodes in the network, they can exchange messages and estimate their *pairwise* drifts relative to other nodes in the network. We define the pairwise drift between nodes $i$ and $j$ as observed at node $i$ as

$$\beta_{j,i}[k] := \beta_j[k] - \beta_i[k] = (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \boldsymbol{\beta}[k]$$

where $\boldsymbol{e}_i \in \mathbb{R}^N$ is a vector of all zeros except for a one in position $i$.

When random edge $(j, i)$ is activated in timeslot $k$, node $s_k = j$ transmits and then node $s_{k-1} = i$ forms the estimate $\hat{\beta}_{j,i}[k]$ of the pairwise drift $\beta_{j,i}[k]$ and subsequently adjusts its local clock drift through a correction

$$\begin{aligned} \beta_i[k+1] &= \beta_i[k] + \mu \hat{\beta}_{j,i}[k] \\ &= \beta_i[k] + \mu (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \boldsymbol{\beta}[k] \end{aligned} \tag{2}$$

where $\mu > 0$ is a stepsize parameter, and the final equality results from our assumption that estimates are perfect. All other local clock drifts in the network (including node $s_k = j$) are not updated, i.e.

$$\beta_\ell[k+1] = \beta_\ell[k] \qquad \forall \ell \neq i. \tag{3}$$

For every $(j, i) \in E$, we can define

$$\boldsymbol{R}_{i,j} := \boldsymbol{e}_i (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \tag{4}$$

and the associated random matrix $\boldsymbol{R}[k]$ having $\mathrm{Prob}[\boldsymbol{R}[k] = \boldsymbol{R}_{i,j}] = p_{i,j}$. Putting (2) and (3) together, we can represent the drift vector update in timeslot $k$ as

$$\begin{aligned} \boldsymbol{\beta}[k+1] &= \boldsymbol{\beta}[k] + \mu \boldsymbol{e}_i (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \boldsymbol{\beta}[k] \\ &= (\boldsymbol{I}_N + \mu \boldsymbol{R}_{i,j}) \, \boldsymbol{\beta}[k] \end{aligned}$$

given that edge $(j, i)$ is activated. Since the active edge is random, the drift update vector in the absence of conditioning becomes

$$\boldsymbol{\beta}[k+1] = \boldsymbol{W}[k] \boldsymbol{\beta}[k] \tag{5}$$

where $\boldsymbol{W}[k] := \boldsymbol{I}_N + \mu \boldsymbol{R}[k]$.

### B. Step 2: Offset Compensation

While compensating for oscillator drift *syntonizes* the nodes in the network, it is not sufficient for *synchronization* because the fixed clock offsets among the nodes in the network are not corrected. The offset vector is defined as

$$\boldsymbol{\Delta}[k] := [\Delta_1[k], \ldots, \Delta_N[k]]^\top \in \mathbb{R}^N$$

and the pairwise offset between nodes $i$ and $j$ as observed at node $i$ is further defined as

$$\Delta_{j,i}[k] := \Delta_j[k] - \Delta_i[k] = (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \boldsymbol{\Delta}[k].$$

To correct these offsets, we assume that the pairwise drift between nodes $i$ and $j$ is negligible. Again referring to Fig. 1, we see that the regular network traffic results in an implicit bidirectional message exchange between node $s_{k-1} = i$ and node $s_k = j$. Node $s_{k-1} = i$ can disambiguate its pairwise clock offset with node $s_k = j$ from the propagation delay $\psi_{i,j} = \psi_{j,i}$. The sender/receiver protocol [14], as shown in Fig. 2, is one example of how this can be achieved. Given a packet transmitted by node $i$ in local time $t_i^{(a)}$, it arrives at node $j$ in local time $t_j^{(b)} = t_i^{(a)} + \psi_{i,j} + \Delta_j - \Delta_i$. The implicit response from node $j$ contains the local timestamps $t_j^{(b)}$ and $t_j^{(c)}$ and arrives at node $i$ at local time $t_i^{(d)} = t_j^{(c)} + \psi_{i,j} + \Delta_i - \Delta_j$. After receiving the response, node $i$ can compute the pairwise clock offset to node $j$ as

$$\frac{(t_j^{(b)} - t_i^{(a)}) - (t_i^{(d)} - t_j^{(c)})}{2} = \Delta_j - \Delta_i = \Delta_{j,i}.$$

The timestamp-free synchronization protocol [15] is another example of how timing offsets can be estimated through bidirectional message exchanges, except through physical layer characteristics of the transmissions and without the use of timestamps.

Assuming random edge $(j, i)$ is activated, node $i$ forms the estimate $\hat{\Delta}_{j,i}[k]$ of the pairwise offset $\Delta_{j,i}[k]$ and adjusts its local clock offset

$$\Delta_i[k+1] = \Delta_i[k] + \mu \hat{\Delta}_{j,i}[k] \tag{6}$$

where $\Delta_i[k]$ is the offset with respect to reference time of node $i$ at time $k$ and $\mu > 0$ is a stepsize parameter. All other local clock offsets in the network are not updated. The offset vector update then follows as

$$\begin{aligned} \boldsymbol{\Delta}[k+1] &= \boldsymbol{\Delta}[k] + \mu \boldsymbol{e}_i (\boldsymbol{e}_j - \boldsymbol{e}_i)^\top \boldsymbol{\Delta}[k] \\ &= (\boldsymbol{I}_N + \mu \boldsymbol{R}_{i,j}) \, \boldsymbol{\Delta}[k] \end{aligned}$$

given that edge $(j, i)$ is activated with $\boldsymbol{R}_{i,j}$ defined in (4). In the absence of conditioning we replace $\boldsymbol{R}_{i,j}$ with $\boldsymbol{R}[k]$ giving

$$\boldsymbol{\Delta}[k+1] = \boldsymbol{W}[k] \boldsymbol{\Delta}[k]$$

with $\boldsymbol{W}[k] := \boldsymbol{I}_N + \mu \boldsymbol{R}[k]$ and $\mathrm{Prob}[\boldsymbol{R}[k] = \boldsymbol{R}_{i,j}] = p_{i,j}$.

In this context, offset compensation is conceptually identical to drift compensation as discussed in Section III-A. The offset update equation has the same form as the drift update equation in (5).

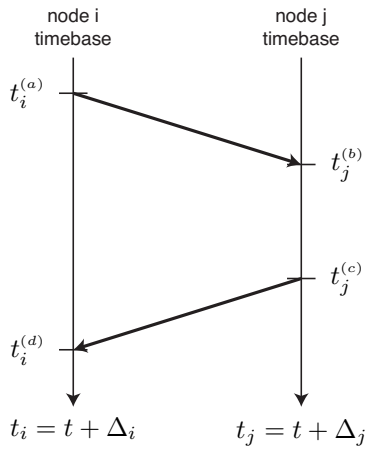


Fig. 2. Bidirectional message exchange for $s_{k-1} = i, s_k = j$.

## IV. Numerical Results

In this section, we investigate the performance of the proposed synchronization approach through simulation, and show that for suitable choice of stepsize, the network exhibits monotonic mean squared convergence of both drifts and offsets toward a consensus clock. We use a "distance from consensus" metric [16]–[18] as a measure of the overall network pairwise drift and offset alignment at time $k$. Defining the mean drift at time $k$ as $\bar{\beta}[k] := \frac{1}{N}\sum_{i=1}^{N}\beta_i[k]$ the distance from consensus metric is then defined as

$$d[k] := \frac{1}{N}\|\boldsymbol{\beta}[k] - \mathbf{1}_N\bar{\beta}[k]\|_2^2.$$

Note that smaller values of $d[k]$ correspond to closer overall synchronization in the network. Also note that the distance from consensus of the offsets $\boldsymbol{\Delta}[k]$ is defined identically with $\boldsymbol{\beta}[k]$ replaced by $\boldsymbol{\Delta}[k]$.

The numerical results in this section assume a network with $N = 10$ nodes, i.i.d. Gaussian distributed initial clock offsets $\Delta_i[0]$ with standard deviation 5 ms, and i.i.d. Gaussian distributed initial drifts $\beta_i[0]$ with standard deviation 100 $\mu$s/iteration, for $i = 1, \ldots, N$. In iterations $k = 0, \ldots, 99$, no synchronization updates occur. During this time, the pairwise drifts remain constant and the pairwise offsets tend to grow. For iterations $k = 100, \ldots, 499$, the drift compensation algorithm runs with randomly transmitting nodes with probabilities specified by the Markov chain $\boldsymbol{P}$. For iterations $k = 500, 501, \ldots$ the offset compensation algorithm runs, also with randomly transmitting nodes with probabilities specified by $\boldsymbol{P}$. In the following, we consider two network configurations: (i) a fully-connected network where all nodes are equally likely to transmit and (ii) a network with a deterministic "round-robin" transmission schedule so that the node transmitting at time $k$ is given by $s_k = \mathrm{mod}(k, N) + 1$ where $\mathrm{mod}(k, N)$ denotes $k$ modulo $N$. Where relevant, we compare behavior of the proposed implicitly acknowledged synchronization algorithm with the explicit acknowledgment algorithm in [10].

### A. Equiprobable Transmissions

In this example, $p_{i,j} = \frac{1}{N-1} = \frac{1}{9}$ for all $i \neq j$ so all nodes are equally likely to transmit in a given time slot. Fig. 3 shows the distance from consensus metrics for both drift and offset, averaged over 1000 Monte-Carlo realizations. In this figure, solid lines represent the proposed consensus synchronization algorithm with implict acknowledgment, whereas dotted lines show the performance of the algorithm presented in [10] which requires explicit acknowledgment of every transmission and consequently more overhead. These results numerically suggest monotonic mean squared convergence of the proposed synchronization algorithm for fixed stepsizes $\mu \in \{0.2, 0.5\}$, and convergence speed appears to be identical to the algorithm with explicit acknowledgments for these stepsizes. Since the proposed algorithm more efficiently exploits the broadcast nature of the wireless medium and does not require explicit acknowledgments, however, the proposed scheme converges twice as fast on a per-transmission basis since the additional overhead of ACKs is not needed. For fixed stepsizes $\mu \in \{1.0, 1.5\}$, the proposed algorithm diverges; in contrast, the algorithm with explicit acknowledgment [10] converges for wider range of stepsizes $\mu \in \{0.2, 0.5, 1.0\}$. This suggests that in the absence of explicit acknowledgment, smaller stepsizes are required for algorithm convergence. Finally, the effect of uncompensated drifts is evident in the first 100 iterations where we see linearly increasing pairwise offsets.

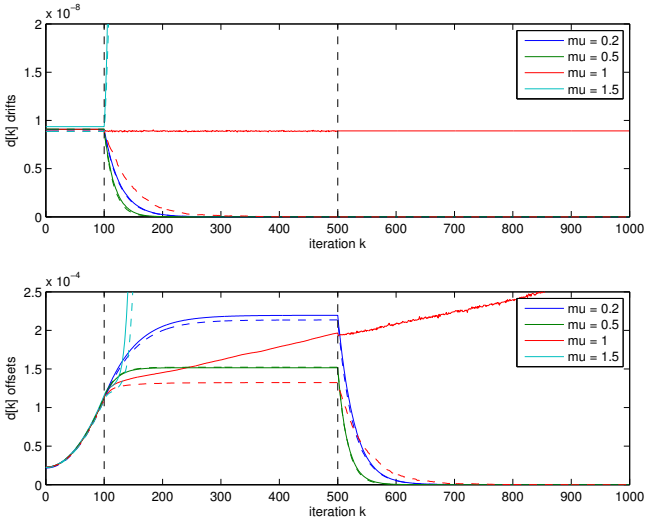


Fig. 3. Empirically averaged distance from consensus metrics for $N = 10$ node synchronization with equiprobable transmit/receive pairs. Solid lines are consensus synchronization via implicit acknowledgment, dotted lines are consensus synchronization with explicit acknowledgment [10].

### B. Deterministic Round-Robin Transmission Schedule

In this example, the $N = 10$ node network operates according to a deterministic round-robin transmission schedule where in each time slot the transmitting node indices follow the pattern $1, 2, \ldots, N, 1, 2, \ldots$. In this case, $\boldsymbol{P}$ is then a circulant shift of the identity matrix so that $p_{i,j} = 1$ for all

$j = i + 1$, and $p_{N,1} = 1$. This choice of $\boldsymbol{P}$ implies that, at the very least, node 2 is connected to nodes 1 and 3, node 1 is connected to nodes 2 and $N$, etc.

Fig. 4 shows the distance from consensus metrics averaged over 1000 Monte-Carlo realizations. We again see lack of convergence when $\mu \in \{1.0, 1.5\}$ and monotonic mean squared convergence for $\mu \in \{0.2, 0.5\}$. One difference in this example with respect to the equiprobable case is that the convergence tends to be slightly slower overall. However, the proposed implicit ACK scheme appears to converge faster than the explicit ACK scheme since the solid lines are uniformly lower than the dotted lines for $\mu \in \{0.2, 0.5\}$.

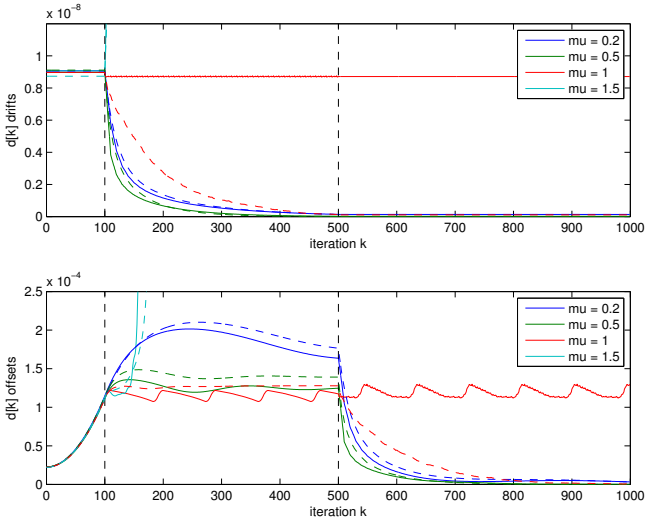


Fig. 4. Empirically averaged distance from consensus metrics for $N = 10$ node synchronization with a deterministic transmission schedule. Solid lines are consensus synchronization via implicit acknowledgment, dotted lines are consensus synchronization with explicit acknowledgment [10].

## V. Conclusions and Extensions

This paper considers a synchronization algorithm that uses existing network traffic and exploits the broadcast nature of the wireless medium to achieve consensus synchronization in a TDD wireless network. Compared to prior work in this area, the proposed scheme does not require explicit acknowledgments, and is therefore applicable to a wider range of network traffic while requiring roughly half the number of transmissions. Numerical results yield the somewhat surprising result that network hierarchy is not necessary and that low-overhead non-hierarchical techniques can exhibit monotonic mean squared convergence to a consensus clock when the local drift/offset updates are sufficiently small. Since convergence to the consensus clock occurs with random node transmissions, these results show that drift and offset consensus can be achieved by gleaning timing estimates from existing network traffic rather than relying on a dedicated synchronization protocol.

Although we have shown that pairwise drift and offset compensation allow the nodes in a wireless network to achieve consensus on the drifts and offsets such that $\beta_i \to \bar{\beta}$ and $\Delta_i \to \bar{\Delta}$ for all $i$, it is worth mentioning that the technique could also be used to synchronize to an external source of reference time if one or more nodes in the network have access to reference time (via, e.g. GPS). The nodes that have access to an external source of reference time simply do not adjust their clock via (2) and (6). This forces the other nodes in the network to adapt to the fixed reference time.

Potential extensions of this work include: (i) convergence analysis for drift and offset compensation, (ii) the development of explicit bounds on the stepsize $\mu$, and (iii) analysis and simulation of non-hierarchical synchronization with stochastic clocks and/or drift and offset estimations errors.

## References


[1] M. Maggs, S. O'Keefe, and D. Thiel, "Consensus clock synchronization for wireless sensor networks," *IEEE Sensors J.*, vol. 12, no. 6, pp. 2269–2277, Jun. 2012.

[2] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[3] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std. 1588, 2008.

[4] W. Lewandowski, J. Azoubib, and W. Klepczynski, "GPS: primary tool for time transfer," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 163–172, Jan 1999.

[5] P. Verissimo, L. Rodrigues, P. V. Issimo, L. I. Rodrigues, and R. A. Redol, "Cesiumspray: a precise and accurate global clock service for large-scale systems," *Journal of Real-Time Systems*, vol. 3, pp. 241–294, 1997.

[6] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul.–Aug. 2004.

[7] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM J. on Appl. Math.*, pp. 1645–1662, 1990.

[8] 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*, vol. 23, no. 5, pp. 1085–1099, May 2005.

[9] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proceedings of the 3rd international conference on Embedded networked sensor systems (ACM SenSys'05)*, November 2-4 2005, pp. 142–153.

[10] D.R. Brown III, A. Klein, and R. Wang, "Monotonic mean squared convergence conditions for pairwise consensus synchronization in wireless networks," *IEEE Transactions on Signal Processing*, in review.

[11] G. Fettweis, E. Zimmermann, V. Jungnickel, and E. Jorswieck, "Challenges in future short range wireless systems," *IEEE Veh. Technol. Mag.*, vol. 1, no. 2, pp. 24–31, Jun. 2006.

[12] E. Baghdady, R. Lincoln, and B. Nelin, "Short-term frequency stability: Characterization, theory, and measurement," *Proc. IEEE*, vol. 53, no. 7, pp. 704–722, Jul. 1965.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, 2006.

[14] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings ACM SenSys 2003*. ACM New York, NY, USA, Nov. 2003, pp. 138–149.

[15] D.R. Brown III and A.G. Klein, "Precise timestamp-free network synchronization," in *Conf. Inf. Sciences and Systems (CISS2013)*, Mar. 2013.

[16] F. Fagnani and S. Zampieri, "Asymmetric randomized gossip algorithms for consensus," in *Proc. IFAC world congress*, 2008, pp. 9051–9056.

[17] ——, "Randomized consensus algorithms over large scale networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 4, pp. 634–649, 2008.

[18] S. Bolognani, R. Carli, and S. Zampieri, "A PI consensus controller with gossip communication for clock synchronization in wireless sensors networks," in *Proc. 1st IFAC Workshop on Estimation and Control of Networked Systems*, 2009.