Distributed Clock Phase and Frequency Synchronization in Half-Duplex TDMA Networks

Itay Zino, Ron Dabora, *Senior Member, IEEE*, and H. Vincent Poor, *Life Fellow, IEEE*

***Abstract*—High clock synchronization accuracy across the nodes in wireless networks is a prerequisite for facilitating high-rate data transmission. Accurate clock synchronization is a particularly challenging goal in networks implementing time division multiple access (TDMA) via half-duplex (HD) communications, as in such networks the updates are temporally sparse, and consequently, clock frequency differences induce significant phase drifts between subsequent updates. Thus, accurate clock synchronization in HD TDMA networks requires synchronizing both clock phases and clock frequencies across the nodes, which is the focus of this work. We consider pulse-coupling (PC)-based distributed clock synchronization, where each node implements its synchronization processing independently, based on its own received clock phases and power measurements. These measurements are then weighted to generate the phase and the frequency correction signals. We first analyze this synchronization framework and motivate decoupling the phase and frequency updates. We then analyze the resulting decoupled structure and derive the asymptotic synchronization accuracy, which is shown to be a function of the weighting coefficients and the unknown propagation delays. This motivates on-line learning of the optimal weights. To that aim, we introduce a novel initialization scheme with unsupervised online training. Simulation results show that the new scheme exhibits excellent synchronization accuracy, which is significantly better than previously proposed schemes, as well as robustness to clock resets and to node mobility.**

***Index Terms*—Distributed synchronization, deep neural network (DNN), unsupervised learning, clock synchronization, time division multiple access (TDMA), pulse coupling (PC).**

I. Introduction

FACILITATING high data rates in wireless networks requires accurate time coordination between the nodes in the network. Generally, clock synchronization schemes implement either packet-coupling or pulse-coupling (PC). In packet-coupling, time-stamps are encoded into the data packets at each node, which are then transmitted to neighboring nodes [1]. The receiving nodes decode the packets to obtain the time stamps, based on which they update their own clocks. While packet-based synchronization has received significant attention, [2], [3], [4], such schemes generally suffer from random processing delays at the nodes and high-power consumption due to the required processing, which makes them less suitable for facilitating low-power, low-complexity accurate time synchronization [5]. In PC-based synchronization, [5], [6], the nodes transmit unique signature sequences. The receiving nodes use the signatures to identify the senders, extract the timing information from the signatures' receive-time stamps and estimate the receive signatures' powers. The nodes then process their receive-time stamps and receive powers to adapt their clocks.

Another critical consideration in synchronization schemes is whether measurement processing is *distributed* or *centralized*. In centralized processing the information from all nodes is jointly processed at a fusion center, which facilitates computation of optimal clock updates. Centralized processing can utilize larger computational resources by designated nodes with appropriate processing power.

As an example, in [7] a global least squares skew and offset estimation was proposed. The main weaknesses of centralized processing are the communications overhead, delays, and power consumption associated with transporting timing information across the network. One approach to decreasing the required resources and overhead is through synchronizing among a small number of users at each time. For example, for long range (LoRa) networks, a synchronization scheme between the gateway and the nodes, where processing is done at the gateway, was proposed in [8], along with a lightweight version which could be applied to synchronization between end nodes. The computational aspects associated with centralized processing are considerably simplified when applying distributed processing in which each node processes only its local information. Here, there are two main approaches: Synchronization with the clock of a single time source or simultaneous synchronization among the clocks. A simple realization of the former is synchronizing with the clock of the Global Positioning System (GPS). Additional approaches for synchronization to a single clock include Reference Broadcast Synchronization (RBS) [9] and the message flooding protocol [10]. Yet, in many applications of distributed synchronization for ad-hoc wireless networks, it is preferable to avoid relying on a single node, namely the nodes operate independently to reach a consensus on the network's clock period and offset among the nodes.

Received 9 April 2025; revised 26 July 2025; accepted 26 August 2025. Date of publication 4 September 2025; date of current version 23 December 2025. The work of I. Zino and R. Dabora was supported in part by the Israel Science Foundation under Grant 584/20 and by the Israel Ministry of Economy via the 5G-WIN Consortium. The work of H. V. Poor was supported in part by an Innovation Grant from Princeton NextG, and by the U.S National Science Foundation under Grants CNS-2128448 and ECCS-2335876. The associate editor coordinating the review of this article and approving it for publication was J. Kang. *(Corresponding author: Ron Dabora.)*

Itay Zino and Ron Dabora are with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Be'er Sheva 8410501, Israel (e-mail: itayzin@post.bgu.ac.il; daborona@bgu.ac.il).

H. Vincent Poor is with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

Digital Object Identifier 10.1109/TCOMM.2025.3606416

A third important factor that affects the performance of synchronization algorithms is whether the network's channel access scheme is *full-duplex (FD)* or *half-duplex (HD)*. In FD access, nodes can transmit and receive simultaneously. Thus, when FD is implemented, the nodes obtain frequent timing measurements, thus, the phase deviations between updates, induced by clock frequency differences, are very small, facilitating synchronization by correcting only the clocks' phases, leaving the clocks' frequencies unsynchronized, e.g., [5], [11], [12]. However, implementing FD access requires specialized design of the transmission and reception circuits as well as dedicated processing, which may significantly impact the cost of the transceivers at the nodes. In contrast, in HD access, as only a single node transmits at any given time, transceiver design is simpler, which translates into low-cost units. However, the temporal sparsity of the clock measurements in HD access requires synchronizing both the clocks' phases and the clocks' frequencies at the nodes, in order to maintain small clock phase deviations between updates. In this work we study clock synchronization in networks operating subject to a strict resources budget. We consider distributed PC-based global clock synchronization for HD time division multiple access (TDMA) networks, which have a wide range of applications including aerial networks [13], cooperative navigation in wireless networks [14], sensor networks [15], automotive radar [16], and distributed antenna arrays [17].

A. Related Work

A common approach for distributed clock synchronization in wireless networks is based on implementing a phase-locked loop (PLL) at each node, as considered in [5] and [12] for FD channel access. In this approach, clock phase measurements are weighted and filtered to generate a phase correction signal for the node's clock. PLL-based clock synchronization was also considered in our previous work, [11], which focused on FD access using *only clock phase corrections*. The work [11] proposed a model-based learning scheme, which uses the loop structure while learning the loop weights, resulting in a dramatically improved performance compared to [5] and [12]. Differently from these works, in the current work we focus on HD communications and show that it requires implementing both period and phase updates. The work [17] considered frequency synchronization in the HD regime with randomly transmitting nodes, for which a simple algorithm was proposed without presenting performance analysis. Previous works that have considered distributed synchronization of *both clocks' frequencies and phases* are essentially based on modeling the clocks as affine functions: In [18] an interactive algorithm for skew and offset estimation based on belief propagation was proposed and analyzed, where each pair of nodes first exchanges a fixed number of time-stamp messages, and subsequently distributed estimation is applied. The work [19] proposed to iteratively eliminate the phase offsets while driving all skews to the same value, and [20] proposed estimating the skews and offsets via a least-squares approach. Note that [19] considered packet-coupling; however, following, e.g., [21], this scheme can also be applied based on pulse timings. Moreover, as the algorithm assumes that each node measures interference-free time stamps from the other nodes, the algorithm implicitly assumes HD operation, [19], [21]. We also note that [19], [21] assumed that the transmissions used for synchronization take place within a relatively small time interval, while a TDMA access regime, as considered in the current work, introduces large delays which can result in poor synchronization accuracy. Another related aspect is the analytical performance derivation of synchronization algorithms operating in the HD regime. We note that the analysis of consensus algorithms typically assumes full-duplex operation and considers first-order filtering, e.g., [22]. Analysis of distributed least-squares with coordinate descent is provided in [23], which converges to the optimal least-squares solution when there are no propagation delays.

In the current work we consider *PLL-based* synchronization of both *clocks' phases and frequencies* via a distributed PC-based scheme. We address both performance analysis and algorithmic implementation. In the following, we elaborate on these contributions.

B. Main Contributions and Organization

Main Contributions: We methodically study PLL-based distributed clock synchronization in HD TDMA networks. The contributions encompass three aspects: (1) Algorithmic structure, (2) analytical performance derivation, and (3) machine learning (ML)-aided implementation. In the context of the *algorithmic structure* we first demonstrate that for HD TDMA networks, both clock phases and periods have to be synchronized to maintain small clock phase differences across all TDMA slots. We next establish *analytically* that when the phase and period updates are not decoupled, a zero error stationary point does not exist, hence, the phase updates have to be decoupled from the frequency updates in order to achieve synchronization of both clock periods and phases. We conclude the algorithmic structure contribution with a proposal of a nested loop structure implementing cyclic decoupled phase and period updates across three TDMA frames, based on weighting the measured phase offsets.

The *analytic contribution* of this work is the performance analysis of a nested synchronization loop in which the processing rate is slower than the incoming symbol rate and the updates are decoupled. It is shown that a nested loop can perfectly synchronize the period even in the presence of propagation delays and low-rate updates, however, the unknown propagation delays induce an inherent clock phase error which depends on the values of the propagation delays as well as the weights. It is clarified that as the propagation delays are unknown at the nodes, *it is not possible to a-priori determine the optimal weights*. This conclusion motivates our third contribution, in which we propose an online weights determination scheme.

Specifically, *the contribution regarding ML implementation* includes the proposal of a low-complexity deep neural network (DNN)-aided weights computation which is trained online, locally, after deployment, in an unsupervised manner, with a novel initialization procedure. As such, training accounts for the propagation delays which were shown in our analysis to be the inherent limiting factor in distributed clock synchronization. Comparing the performance of the proposed scheme with the state-of-the-art we observe the significant gains offered by our proposed novel approach.

Organization: The rest of this work is organized as follows: Sec. II details the clock model and the network setup. Sec. III presents the nested loop structure with decoupled updates, and Sec. IV analytically derives its asymptotic performance as a function of the weights and the propagation delays. Sec. V details the proposed DNN-aided algorithm and the associated training scheme. Section VI presents simulation tests and discussion. Lastly, Section VII concludes the work.

Notation: Let $\mathcal{R}$ and $\mathcal{Z}$ denote the sets of real numbers and of integers, respectively. We denote vectors with boldface letters, e.g., $\mathbf{X}$ and sets with calligraphic letters, e.g., $\mathcal{X}$. $x \sim \mathbb{U}[a, b]$ denotes that x was selected according to a uniform distribution over the interval $a \leq x \leq b$, $a, b \in \mathcal{R}$, and $\mathbb{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2. We use $(k \bmod N) \equiv ((k))_N$ to denote the modulo N operation. We use I_N and $\mathsf{0}_N$ to denote the $N \times N$ identity matrix and the $N \times N$ all-zero matrix, respectively. Square brackets denote the discrete-time (DT) indices, $X \leftarrow Y$ denotes that Y is stored in X, and $'\backslash'$ denotes the set difference operand.

II. Preliminaries: Distributed Pulse-Coupled Time Synchronization for Wireless Networks

A. Network and Clock Models

We consider a network with N nodes, indexed by $i \in \{1, 2, \ldots, N\} \triangleq \mathcal{I}_N$. Each node i has its own clock. As the frequency of a clock oscillator can vary within a limited range, then, w.l.o.g. we define a reference period T_{com}, and measure the periods of the clocks in the network w.r.t. T_{com}. At time-slot interval $k \in \mathcal{Z}^+$, the period of node i's clock is expressed as $T_i^{\text{prd}}[k] = T_{\text{com}} + T_i[k]$, where $T_i[k]$ denotes the clock period offset w.r.t. the common reference T_{com} at time-slot interval $k \geq 0$, at node i. Using the common assumption of ignoring phase noise, e.g., [5], the clock time at node i, $\phi_i[k]$, also referred to as the *clock phase*, can be expressed as

$$\phi_i[k+1] = \phi_i[k] + T_{\text{com}} + T_i[k], \qquad k \geq 0. \tag{1}$$

In this model, $\phi_i[0]$ and $T_i[0]$ denote the clock time and clock period offset at node i at startup. This model appropriately represents the clock tick times which determine the transmission times in HD TDMA wireless networks. Note that the clock values are real-valued, and the discrete index k represents the time slot interval number. After startup, the nodes begin transmitting their signatures. In accordance with the HD TDMA operation, at the k-th transmission interval, node $i = ((k))_N + 1$ transmits its signature at time $\phi_i[k]$ and the remaining $N - 1$ nodes receive.

Each receiving node assigns a time stamp and a received signal power value to each received signature. Consider reception at node $j \neq i$: Let $q_{j,i}$ denote the propagation delay between nodes i and j. This propagation delay is *unknown* at the nodes. The receive time stamp node j assigns to node i's transmission at the k-th time interval, s.t., $i = ((k))_N + 1$, is the time node

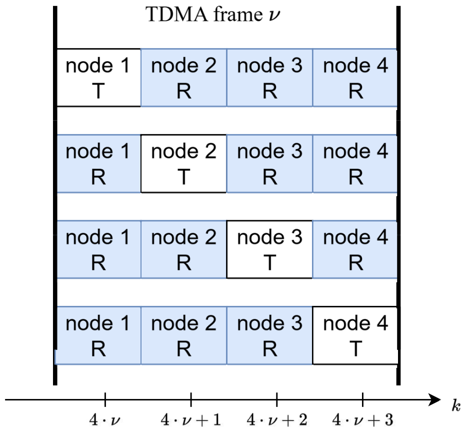


Fig. 1. TDMA update cycle with $N = 4$: The white blocks (also denoted with 'T') mark the transmitting node whose clock times are measured at the receiving nodes (shaded blocks denoted with 'R') at the ν-th TDMA frame. The node index indicates its order within the TDMA transmission cycle.

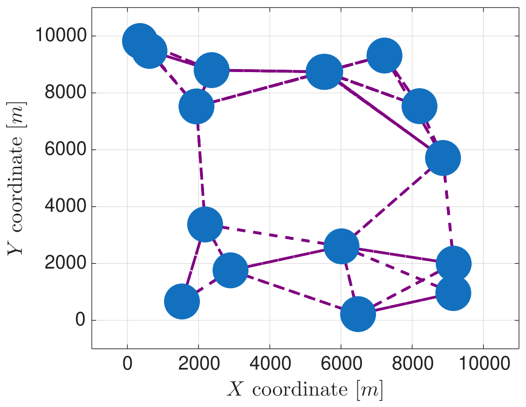


Fig. 2. Node placement for the representative scenario. Nodes are denoted by circles and links over which the signal is received above the reception threshold are denoted by lines. Parameters are detailed in Footnote II-B.

j observed the k-th transmission relative to the time at node j at the k-th interval, i.e., with $i = ((k))_N + 1$, node $j \neq i$ measures node i's time stamp as

$$\Delta t_{j,i}[k] = \phi_i[k] + q_{j,i} - \phi_j[k].$$

We emphasize that $\Delta t_{j,i}[k]$ is a *measured* quantity at node j and *not a computed* quantity, since both $\phi_i[k]$ and $q_{j,i}$ are unknown at node j. In addition to $\Delta t_{j,i}[k]$, each receiver also measures the received signature's power. Let $P_{j,i}[k]$ denote the power of the signature from node i measured at node j. Thus, the overall information collected by node j during the ν-th TDMA frame is the set $\mathcal{M}_j(\nu) \triangleq \{P_{j,i}[k], \Delta t_{j,i}[k]\}_{k=\nu N, ((k))_N \neq j-1}^{\nu(N+1)-1}$. This is demonstrated with $N = 4$ in Fig. 1, where the white block marks the node whose signature is observed at the shaded nodes during the ν-th TDMA frame.

In the HD regime, clock correction is applied once in a TDMA frame. In this work we assume the updates are applied at the end of a TDMA frame, i.e., after all nodes completed a transmissions cycle since their last update, which occurs at time interval k s.t. $((k))_N = N - 1$. At these k indices each node $j \in \mathcal{I}_N$ independently processes its received information, $\mathcal{M}_j(\nu)$, to generate correction signals applied to update its TDMA clock period and clock phase. In this work we consider *strongly connected* network configurations [12], in which there is a path between every pair of nodes in the network, possibly going through intermediate nodes, see Fig. 2.

B. Scenario Parameters and Performance Measures

We use scenario parameters that represent practical deployments. We set the network size to $N = 16$ nodes, deployed in a square area with a side-length of 10 [Km]. We assume 2-ray propagation where each node has an antenna whose height is 1.5 [m], transmit power of 33 [dBm], and a reception threshold of $P_{\text{th}} = -114$ [dBm], [24], [25], [26]. To visualize the impact of these parameters of the network, Fig. 2 depicts an instance of a randomly generated node deployment scenario[1] in which 30% of the links carry signals that are received at their respective destinations above the reception threshold. These links, referred to as the *active links*, are marked by the lines in Fig. 2. The network in Fig. 2 is referred to as the *representative scenario*.[2] The performance in Sec. VI is reported using statistics based on 800 network realizations, obtained by randomly and uniformly placing nodes within the network area, verifying that about 30% of the links are active. For generating the clocks at the nodes, the nominal duration of the time-slot is set to $T_{\text{nom}} = 5$ [msec], and the respective nominal TDMA frequency is $f_{\text{nom}} = 200$ [Hz]. As in [24], we model clock uncertainty by generating the initial clock frequency f_i at each node i randomly, according to a uniform distribution $f_i \sim \mathbb{U}\left[\frac{1}{T_{\text{nom}}}(1 - 150 \cdot 10^{-6}), \frac{1}{T_{\text{nom}}}(1 + 150 \cdot 10^{-6})\right]$, which corresponds to a clock accuracy of 150 [ppm]. The initial TDMA slot period at node i is obtained as $T_i^{\text{prd}}[0] = \frac{1}{f_i}$. Each node is then assigned a randomly generated initial phase $\phi_i[0]$, $\phi_i[0] \sim \mathbb{U}\big[0, T_i^{\text{prd}}[0]\big]$.

1) Performance Measures: We define the normalized phase difference (NPD) at node i, denoted by $\text{NPD}_i[k]$, to be the difference between the clock phase at node i and the clock phase at node 1, divided by the instantaneous mean period, denoted by $\bar{T}[k] \triangleq \frac{1}{N}\sum_{i=1}^{N} T_i^{\text{prd}}[k]$:

$$\text{NPD}_i[k] = \big(\phi_i[k] - \phi_1[k]\big)/\bar{T}[k]. \tag{2}$$

Network synchronization accuracy is measured via the NPD range (NPDR), which is the maximal clock phase difference

[1]The coordinates of the nodes, the initial periods, and the initial phases for the representative scenario are $\{(x_i[\text{Km}], y_i[\text{Km}], \phi_i[0], T_i[0])\}_{i=1}^{16} = \big[(6.015, 2.6, 0.0021, 0.049996285), (2.366, 8.8, 0.0014, 0.049998529), (1.94, 7.523, -0.0011, 0.050004372), (9.151, 0.959, -0.0007, 0.049998932), (5.51, 8.741, 0.0004, 0.049995185), (7.221, 9.313, -0.0003, 0.049998797), (8.205, 7.532, 0.0013, 0.050001951), (0.62, 9.476, -0.0020, 0.049994327), (2.183, 3.372, -0.0004, 0.050000753), (5.556, 8.739, -0.0021, 0.050007868), (2.896, 1.758, 0.0014, 0.050005321), (8.865, 5.709, -0.0016, 0.049995986), (1.532, 0.663, 0.0006, 0.050005278), (9.165, 1.992, -0.0023, 0.050005455), (6.479, 0.218, 0.001, 0.049994403), (0.354, 9.823, 0.0022, 0.049996557)\big]$.

[2]This scenario is used only for illustrating phase trajectories throughout the manuscript. Note that our conclusions are drawn based on analysis and statistical simulations, and do not rely on a single scenario.

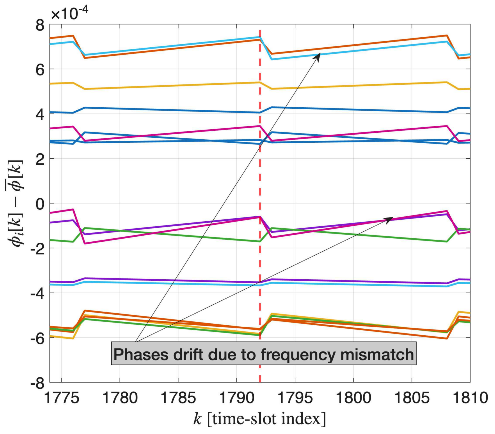


Fig. 3. Clock phase trajectories for the classic algorithm [5] (synchronizing only the clock phase while the period remains fixed) tested for the representative HD TDMA deployment of Fig. 2. Each line depicts the clock phase trajectory at one node out of the $N = 16$ nodes.

between any pair of clocks in the network, normalized to the instantaneous mean period, $\bar{T}[k]$:

$$\text{NPDR}[k] \triangleq \Big(\max_{i_1 \in \mathcal{I}_N} \phi_{i_1}[k] - \min_{i_2 \in \mathcal{I}_N} \phi_{i_2}[k]\Big)/\bar{T}[k]. \tag{3}$$

As the NPDR is a normalized quantity that represents the phase spread relative to the nominal period, hence, it is unitless.

III. The Algorithmic Structure: Decoupled Nested Updates for HD TDMA Networks

We focus on distributed algorithms operating independently at each node, updating the clock at the node based on local measurements.

A. Mitigating Phase Drift via a Nested Phase and Frequency Synchronization Loop

As mentioned in Sec. I, the classic scheme of [5] operates to achieve network clock synchronization by updating only the clocks' phases, leaving the periods of the clocks at the nodes unsynchronized. Applying the classic scheme [5] to networks employing HD TDMA access, where phase correction is applied at the end of a TDMA frame, results in clock phase drifts between corrections. To visualize these phase drifts, we begin by recalling the classic scheme [5]: Define the *neighborhood set* of node i, $\mathcal{N}(i)$, to contain all the nodes $j \in \mathcal{I}_N$, $j \neq i$, whose signatures are received at node i with power higher than P_{th}. As the scheme in [5] does not vary the clocks' periods, they remain fixed at $\{T_i^{\text{prd}}[0]\}_{i=1}^{N}$. The clock update at node i at time-slots $k = \nu N$, $\nu \in \mathcal{Z}^+$ is now given as (see [5, Eqn. (16)])

$$\phi_i[k+1] = \phi_i[k] + T_i^{\text{prd}}[0] + \varepsilon_0 \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \cdot \Delta t_{i,j}[k+j-1], \tag{4}$$

where ε_0 denotes the *loop gain*, and the $\alpha_{i,j}$'s are predetermined coefficients set as in [5, Eqn. (8)]; see also Sec. IV-B.1. At $k \neq \nu N$, the clock update is $\phi_i[k+1] = \phi_i[k] + T_i^{\text{prd}}[0]$. Fig. 3 depicts sample clock phase trajectories obtained with the classic algorithm (4) applied in the HD regime, for the representative network detailed in Fig. 2 with parameters specified in Footnote II-B. The clock phase values are plotted w.r.t. to the mean instantaneous phase, defined as $\bar{\phi}[k] \triangleq \frac{1}{N}\sum_{i=1}^{N}\phi_i[k]$. It is observed that the lack of frequency synchronization induces significant phase drifts across the nodes within a TDMA frame.

A straightforward approach for achieving clock phase and frequency synchronization in HD networks is to generalize the model of [5] by *adding a nested period synchronization loop for synchronizing* $\{T_i[k]\}_{i=1}^{N}$ *across the nodes*, possibly using different weights for the period loop than those in the phase loop. Extending the algorithm (4) to include period updates is now implemented as follows: At startup, node i initializes its storage variables for the period, phase, and power, respectively, $\Delta_T^{(i)}[j] = \Delta_\phi^{(i)}[j] = P^{(i)}[j] = 0,\ \forall j \in \mathcal{I}_N \setminus i$. At time interval k, node $j = ((k))_N + 1$ transmits and nodes $i \in \mathcal{I}_N \setminus j$ receive. Then, for any node $i \in \mathcal{I}_N \setminus j$ for which $P_{i,j}[k] > P_{\text{th}}$, the node first updates $\mathcal{N}(i)$ to include j, and then computes and stores (in the order stated)

$$\begin{aligned}
&\Delta_T^{(i)}[j] \leftarrow \left(\Delta t_{i,j}[k] - \Delta_\phi^{(i)}[j]\right)/N\\
&\Delta_\phi^{(i)}[j] \leftarrow \Delta t_{i,j}[k]\\
&P^{(i)}[j] \leftarrow P_{i,j}[k].
\end{aligned}$$

Observe that $\Delta_T^{(i)}[j]$ represents the *difference between the clock periods* at nodes i and j, and $\Delta_\phi^{(i)}[j]$ is the *difference between the clock phases* at these nodes. The nested scheme is described by the following clock update equations:

$$\phi_i[k+1] = \phi_i[k] + T_{\text{com}} + T_i[k] + \Omega_i[k] \tag{5a}$$
$$T_i[k+1] = T_i[k] + \Delta T_i[k], \tag{5b}$$

where the corrections $\Omega_i[k]$ and $\Delta T_i[k]$ are computed by weighting the respective differences:

$$\Delta T_i[k] = \begin{cases} \Delta T_i[k-1], & ((k))_N \neq N-1 \\ \frac{\varepsilon_T}{N}\sum\limits_{j\in\mathcal{N}(i)} \alpha_{i,j}^{(T)} \cdot \Delta_T^{(i)}[j], & ((k))_N = N-1 \end{cases} \tag{6a}$$

$$\Omega_i[k] = \begin{cases} 0, & ((k))_N \neq N-1 \\ \varepsilon_\phi \sum\limits_{j\in\mathcal{N}(i)} \alpha_{i,j}^{(\phi)} \cdot \Delta_\phi^{(i)}[j], & ((k))_N = N-1 \end{cases} \tag{6b}$$

In (6), $(\varepsilon_T, \alpha_{i,j}^{(T)})$, $(\varepsilon_\phi, \alpha_{i,j}^{(\phi)})$ denote the loop gains and weighting coefficients for the period correction loop and the phase correction loop, respectively. Observe that in HD the correction is applied *once in a TDMA frame* while in FD the correction is applied at every time-slot, which is N times faster than in HD. Note that period correction is applied gradually over a TDMA frame duration to avoid a large instantaneous phase jump, which is harder to track.

The Appendix analyzes the frequency synchronization error after convergence when clock phase and clock period updates are simultaneously applied, as in (6). The analysis shows that simultaneous updates result in inherent clock frequency differences between the nodes after convergence: Even if all clocks are set to the same frequency, the *propagation delays* induce a non-zero correction signal at the period update loop which shifts the frequencies away from the zero-error point. Moreover, when the phase updates are applied simultaneously with the frequency updates then the clock phase updates may compensate for clock frequency errors. This can be directly deduced from [5] where clock phase synchronization is achieved while clock frequency differences exist (but without propagation delays) while the loop updates only the clock phases but not the clock frequencies. Thus, *a direct implementation of a nested period correction loop* will result in clock period differences remaining after convergence, which implies that the clock phase drift problem demonstrated in Fig. 3 will not be resolved.

B. Decoupling Clock Phase and Clock Period Updates

To avoid the negative effect of coupling between the phase and the frequency updates, we propose a *decoupled nested synchronization* scheme. This scheme, referred to in [24] as the extended Simeone-Spagnolini-BarNess-Strogatz algorithm (ESSBSA), uses the update rule (5), but decouples the computation of the phase and period correction signals to facilitate convergence of both clock periods and clock phases. To achieve this decoupling, the updates are applied periodically in a sequence of 3 alternating actions, and therefore, a cycle of period and phase updates spans $3N$ time slots, equivalent to 3 TDMA frames. We denote the interval for which $((k))_{3N} \in \{0, 2N-1\}$ as a "data collection" interval, at which the clocks are updated via Eqns. (5) where the correction terms are set to $\Omega_i[k] = \Delta T_i[k] = 0$. Hence, the collection interval consists of two TDMA frames, where at the each TDMA frame, node i collects features as described above. At the end of the collection interval each node $i \in \mathcal{I}_N$ obtains $\Delta_T^{(i)}[j]$, $\Delta_\phi^{(i)}[j]$, and $P^{(i)}[j]$ for all $j \in \mathcal{N}(i)$. Next, when $((k))_{3N} \in \{2N-1, 3N-2\}$, a "period update" action is applied with a period correction signal $\Delta T_i[k]$ given by

$$\Delta T_i[k] = \begin{cases} \frac{\varepsilon_T}{N}\sum\limits_{m\in\mathcal{N}(i)} \alpha_{i,m}^{(T)} \cdot \Delta_T^{(i)}[m], & ((k))_{3N} = 2N-1 \\ \Delta T_i[k-1], & 2N \leq ((k))_{3N} \leq 3N-2 \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Finally, when $((k))_{3N} = 3N-1$, a "phase update" action is applied with a correction signal $\Omega_i[k]$, computed as

$$\Omega_i[k] = \begin{cases} 0, & ((k))_{3N} \neq 3N-1 \\ \varepsilon_\phi \sum\limits_{m\in\mathcal{N}(i)} \alpha_{i,m}^{(\phi)} \cdot \Delta_\phi^{(i)}[m], & ((k))_{3N} = 3N-1. \end{cases} \tag{8}$$

In Sec. IV-B we recall different assignments of the $\alpha_{i,j}$'s proposed in previous works.

The general synchronization algorithm structure described above is depicted in Fig. 4. The difference between synchronization algorithms using this structure lies in the weights used for generating the correction signals, i.e., the "Phase Correction Signal" block and the "Period Correction Signal" block, which will be defined for each algorithm. Computation of the phase and the period correction signals for the ESSBSA, given in Eqns. (7), (8), are schematically depicted in Fig. 5.

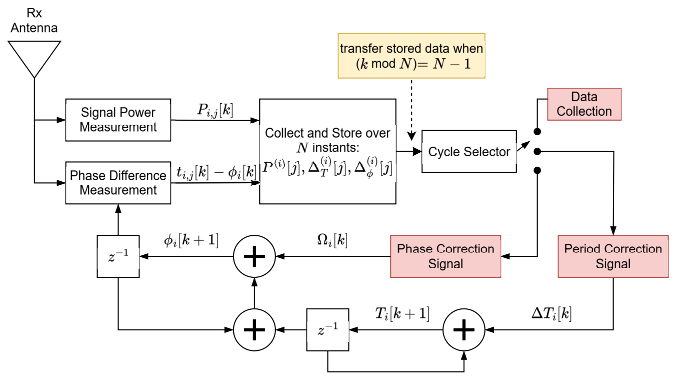


Fig. 4. Schematic description of the nested synchronization algorithm structure with decoupled updates.

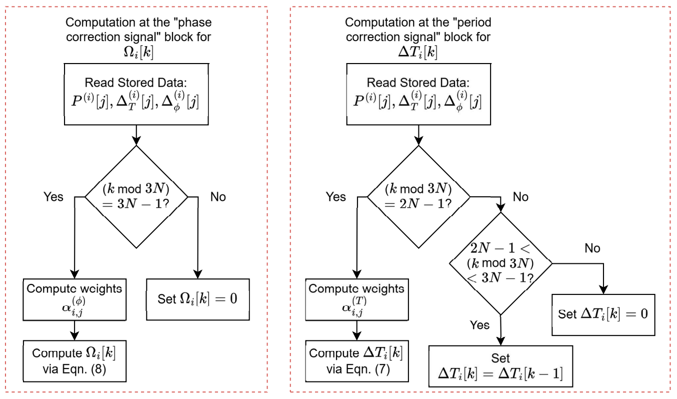

Computation at the "phase correction signal" block for $\Omega_i[k]$
Read Stored Data: $P^{(i)}[j], \Delta_T^{(i)}[j], \Delta_\phi^{(i)}[j]$
$(k \bmod 3N) = 3N-1?$
Yes
No
Compute weights $\alpha_{i,j}^{(\phi)}$
Set $\Omega_i[k]=0$
Compute $\Omega_i[k]$ via Eqn. (8)
Computation at the "period correction signal" block for $\Delta T_i[k]$
Read Stored Data: $P^{(i)}[j], \Delta_T^{(i)}[j], \Delta_\phi^{(i)}[j]$
$(k \bmod 3N) = 2N-1?$
Yes
No
Compute weights $\alpha_{i,j}^{(T)}$
$2N-1 < (k \bmod 3N) < 3N-1?$
No
Yes
Set $\Delta T_i[k]=0$
Compute $\Delta T_i[k]$ via Eqn. (7)
Set $\Delta T_i[k] = \Delta T_i[k-1]$

Fig. 5. Schematic description of the correction signal blocks in Fig. 4 for the ESSBSA algorithm.

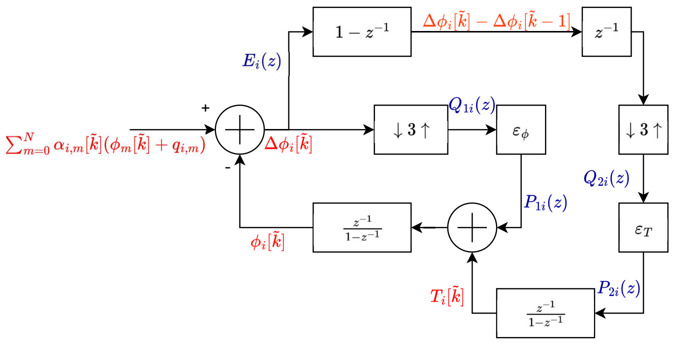

$\sum_{m=0}^{N}\alpha_{i,m}[\tilde{k}](\phi_m[\tilde{k}]+q_{i,m})$
+
-
$E_i(z)$
$1-z^{-1}$
$\Delta\phi_i[\tilde{k}]-\Delta\phi_i[\tilde{k}-1]$
z^{-1}
$\Delta\phi_i[\tilde{k}]$
$\downarrow 3\uparrow$
$Q_{1i}(z)$
ε_ϕ
$\downarrow 3\uparrow$
$Q_{2i}(z)$
$P_{1i}(z)$
ε_T
$\frac{z^{-1}}{1-z^{-1}}$
$\phi_i[\tilde{k}]$
$T_i[\tilde{k}]$
$P_{2i}(z)$
$\frac{z^{-1}}{1-z^{-1}}$

Fig. 6. Schematic diagram of the nested loop operations.

Evidently, different assignments of the weights $\alpha_{i,j}$'s result in different algorithms, with different steady-state NPDR. In Sec. IV we analytically derive the asymptotic NPDR of such algorithms for strongly connected configurations, requiring only that for $\chi \in \{T, \phi\}$ and $i \in \mathcal{I}_N$, it holds that $\sum_{m=1}^{N} \alpha_{i,m}^{(\chi)} = 1$, $\alpha_{i,i}^{(\chi)} = 0$, and $\alpha_{i,m}^{(\chi)} \geq 0$, for all $m \in \mathcal{I}_N$, $m \neq i$.

IV. NPDR Derivation for Decoupled Updates in Distributed Synchronization for TDMA Networks

In this section we analyze the dynamics of our proposed nested loop structure. The challenge and novelty in the analysis follows as the rate of the correction signals is one third

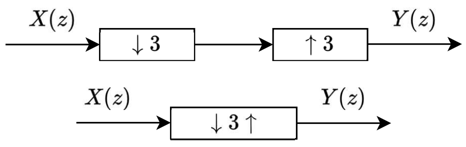


Fig. 7. Downsampling and subsequently zero padding.

of the rate of the TDMA frame, thus a complete clock update cycle spans 3 TDMA frames. Furthermore, the two corrections are applied at two different instants within the clock update cycle. The following analysis shows that these assumptions result in a *linear time-varying system*. As far as we know, such a multi-rate synchronization architecture was not analyzed before. Referring to the loop depicted in Fig. 6 we note that it assumes all clock phase measurements are collected at the end of a frame, while the scheme in Sec. IV-B collects clock phase measurements throughout the frame (see Fig. 1). This assumption simplifies the analysis while capturing the key elements of the nested loop architecture which impact its asymptotic performance. In particular, the decoupled processing is implemented via downsampling by a factor of 3 and immediately zero padding by a factor of 3. The concatenation is marked by the symbol '$\downarrow 3\uparrow$', see Fig. 7. Note that as the scheme in Fig. 6 operates at TDMA frame intervals, then each time interval for the loop represents N time intervals for the synchronization algorithm. Now, consider the block '$\downarrow 3\uparrow$' in Fig. 7: Letting $X(z)$ denote the Z-transform of its input, the Z-transform of the output, $Y(z)$, is given as $Y(z) = \frac{1}{3}\sum_{m=0}^{2} X\left(e^{-j\frac{2\pi}{3}m} z\right)$. Next, we derive the Z-transforms marked along the signal paths in Fig. 6 as follows:

$$Q_{1i}(z) = \frac{1}{3}\sum_{m=0}^{2} E_i\left(e^{-j2\pi\frac{m}{3}} z\right)$$

$$P_{1i}(z) = \frac{\varepsilon_\phi}{3}\sum_{m=0}^{2} E_i\left(e^{-j2\pi\frac{m}{3}} z\right)$$

$$\begin{aligned} Q_{2i}(z) &= \frac{1}{3}\sum_{m=0}^{2}\left(z^{-1}\left(1-z^{-1}\right)E_i(z)\right)\Big|_{z\mapsto e^{-j2\pi\frac{m}{3}} z} \\ &= \frac{1}{3}\sum_{m=0}^{2}\left(e^{j2\pi\frac{m}{3}} z^{-1}\left(1-e^{j2\pi\frac{m}{3}} z^{-1}\right) E_i\left(e^{-j2\pi\frac{m}{3}} z\right)\right) \end{aligned}$$

$$P_{2i}(z) = \frac{\varepsilon_T}{3}\sum_{m=0}^{2}\left(e^{j2\pi\frac{m}{3}} z^{-1} - e^{j4\pi\frac{m}{3}} z^{-2}\right) E_i\left(e^{-j2\pi\frac{m}{3}} z\right)$$

$$T_i(z) = P_{2i}(z)\,\frac{z^{-1}}{1-z^{-1}}$$

The Z-transform of the clock phase signal is now expressed as

$$\begin{aligned} \Phi_i(z) &= \left(T_i(z) + P_{1i}(z)\right)\frac{z^{-1}}{1-z^{-1}} \\ &= \left(P_{2i}(z)\,\frac{z^{-1}}{1-z^{-1}} + P_{1i}(z)\right)\frac{z^{-1}}{1-z^{-1}}. \end{aligned}$$

Substituting the expressions for $P_{1i}(z)$ and $P_{2i}(z)$ into $\Phi_i(z)$ we obtain

$$\begin{aligned}&\left(1-2z^{-1}+z^{-2}\right)\Phi_i(z)\\&=\frac{\varepsilon_T}{3}\sum_{m=0}^{2}\left(e^{j2\pi\frac{m}{3}}z^{-3}-e^{j4\pi\frac{m}{3}}z^{-4}\right)E_i\left(e^{-j2\pi\frac{m}{3}}z\right)\\&\quad+\frac{\varepsilon_\phi}{3}\sum_{m=0}^{2}\left(z^{-1}-z^{-2}\right)E_i\left(e^{-j2\pi\frac{m}{3}}z\right).\end{aligned}$$

Applying the inverse Z-transform yields (we use $\tilde{k}$ to emphasize that the time unit is a TDMA frame and not to a time-slot)

$$\begin{aligned}&\phi_i[\tilde{k}]-2\phi_i[\tilde{k}-1]+\phi_i[\tilde{k}-2]\\&=\frac{\varepsilon_T}{3}\sum_{m=0}^{2}e^{j2\pi\frac{m}{3}(\tilde{k}-2)}\left(e_i[\tilde{k}-3]-e_i[\tilde{k}-4]\right)\\&\quad+\frac{\varepsilon_\phi}{3}\sum_{m=0}^{2}\left(e^{j2\pi\frac{m}{3}(\tilde{k}-1)}e_i[\tilde{k}-1]-e^{j2\pi\frac{m}{3}(\tilde{k}-2)}e_i[\tilde{k}-2]\right).\end{aligned}$$

Observing that

$$\sum_{m=0}^{2}e^{j2\pi\frac{m}{3}\ell}=\begin{cases}3, & ((\ell))_3=0\\0, & \text{otherwise}\end{cases}\triangleq 3\delta[((l))_3],$$

we arrive at the following linear, time-varying system:

$$\begin{aligned}\phi_i[\tilde{k}]&=2\phi_i[\tilde{k}-1]-\phi_i[\tilde{k}-2]\\&\quad+\left(\varepsilon_T\left(e_i[\tilde{k}-3]-e_i[\tilde{k}-4]\right)-\varepsilon_\phi e_i[\tilde{k}-2]\right)\delta[((\tilde{k}))_3-2]\\&\quad+\varepsilon_\phi e_i[\tilde{k}-1]\delta[((\tilde{k}))_3-1]\\&=\begin{cases}0, & ((\tilde{k}))_3=0\\\varepsilon_\phi\, e_i[\tilde{k}-1], & ((\tilde{k}))_3=1\\\varepsilon_T\left(e_i[\tilde{k}-3]-e_i[\tilde{k}-4]\right)-\varepsilon_\phi\, e_i[\tilde{k}-2], & ((\tilde{k}))_3=2.\end{cases}\end{aligned}$$

Next, for $\chi\in\{T,\phi\}$, $i,m\in\{1,2,\ldots N\}$, define the column vectors $\boldsymbol{\alpha}_i^{(\chi)}$, $\mathbf{q}_i$, and $\boldsymbol{\phi}[\tilde{k}]$ s.t.

$$\begin{aligned}\left(\boldsymbol{\alpha}_i^{(\chi)}\right)_m&=\alpha_{i,m}^{(\chi)}, m\neq i,\text{and }\left(\boldsymbol{\alpha}_i^{(\chi)}\right)_i=-1\\\left(\mathbf{q}_i\right)_m&=q_{i,m}, m\neq i,\text{and }\left(\mathbf{q}_i\right)_i=0\\\left(\boldsymbol{\phi}[\tilde{k}]\right)_i&=\phi_i[\tilde{k}].\end{aligned}$$

Recall that $\sum_{m=1}^{N}\alpha_{i,m}^{(\chi)}=1$, $\alpha_{i,i}^{(\chi)}=0$, hence, defining $\mathbf{1}^{(N)}$ to be the $N\times 1$ vector of ones, it holds that $\left(\boldsymbol{\alpha}_i^{(\chi)}\right)^T\mathbf{1}^{(N)}=0$. Letting $\gamma_i^{(\phi)}=\left(\boldsymbol{\alpha}_i^{(\phi)}\right)^T\mathbf{q}_i$ we write

$$\begin{aligned}e_i[\tilde{k}]&=\sum_{m=1}^{N}\alpha_{i,m}[\tilde{k}](\phi_m[\tilde{k}]+q_{i,m})-\phi_i[\tilde{k}]\\&=\left(\boldsymbol{\alpha}_i[\tilde{k}]\right)^T\boldsymbol{\phi}[\tilde{k}]+\left(\boldsymbol{\alpha}_i[\tilde{k}]\right)^T\mathbf{q}_i=\left(\boldsymbol{\alpha}_i[\tilde{k}]\right)^T\boldsymbol{\phi}[\tilde{k}]+\gamma_i[\tilde{k}],\end{aligned}$$

thus we obtain

$$\begin{aligned}\phi_i[\tilde{k}]&=2\phi_i[\tilde{k}-1]-\phi_i[\tilde{k}-2]\\&\quad+\Big(\varepsilon_T\cdot\left(\boldsymbol{\alpha}_i^{(T)}\right)^T\cdot\left(\boldsymbol{\phi}[\tilde{k}-3]-\boldsymbol{\phi}[\tilde{k}-4]\right)\\&\quad-\varepsilon_\phi\cdot\left(\boldsymbol{\alpha}_i^{(\phi)}\right)^T\cdot\boldsymbol{\phi}[\tilde{k}-2]-\varepsilon_\phi\gamma_i^{(\phi)}\Big)\delta[((\tilde{k}))_3-2]\\&\quad+\left(\varepsilon_\phi\cdot\left(\boldsymbol{\alpha}_i^{(\phi)}\right)^T\cdot\boldsymbol{\phi}[\tilde{k}-1]+\varepsilon_\phi\gamma_i^{(\phi)}\right)\delta[((\tilde{k}))_3-1].\end{aligned}$$

Finally, letting

$$\begin{aligned}\mathsf{A}^{(\chi)}&\triangleq[\boldsymbol{\alpha}_1^{(\chi)},\boldsymbol{\alpha}_2^{(\chi)},\ldots,\boldsymbol{\alpha}_N^{(\chi)}]^T\\\boldsymbol{\gamma}^{(\phi)}&=[\gamma_1^{(\phi)},\gamma_2^{(\phi)},\ldots,\gamma_N^{(\phi)}]^T,\end{aligned}$$

we obtain a linear, *time-varying difference vector equation* for the clock synchronization scheme in HD TDMA networks:

$$\begin{aligned}&\boldsymbol{\phi}[\tilde{k}]\\&=\left(2\mathsf{I}_N+\varepsilon_\phi\mathsf{A}^{(\phi)}\delta[((\tilde{k}))_3-1]\right)\boldsymbol{\phi}[\tilde{k}-1]\\&\quad+\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}\delta[((\tilde{k}))_3-1]\\&\quad-\left(\mathsf{I}_N+\varepsilon_\phi\mathsf{A}^{(\phi)}\delta[((\tilde{k}))_3-2]\right)\boldsymbol{\phi}[\tilde{k}-2]\\&\quad-\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}\delta[((\tilde{k}))_3-2]\\&\quad+\varepsilon_T\mathsf{A}^{(T)}\left(\boldsymbol{\phi}[\tilde{k}-3]-\boldsymbol{\phi}[\tilde{k}-4]\right)\delta[((\tilde{k}))_3-2]\\&=\begin{cases}2\boldsymbol{\phi}[\tilde{k}-1]-\boldsymbol{\phi}[\tilde{k}-2], & ((\tilde{k}))_3=0\\\left(2\mathsf{I}_N+\varepsilon_\phi\mathsf{A}^{(\phi)}\right)\boldsymbol{\phi}[\tilde{k}-1]+\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}\\\quad-\boldsymbol{\phi}[\tilde{k}-2], & ((\tilde{k}))_3=1\\2\boldsymbol{\phi}[\tilde{k}-1]-\left(\mathsf{I}_N+\varepsilon_\phi\mathsf{A}^{(\phi)}\right)\boldsymbol{\phi}[\tilde{k}-2]-\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}\\\quad+\varepsilon_T\mathsf{A}^{(T)}\left(\boldsymbol{\phi}[\tilde{k}-3]-\boldsymbol{\phi}[\tilde{k}-4]\right), & ((\tilde{k}))_3=2.\end{cases}\end{aligned}$$

A. Asymptotic Error Analysis in the HD Regime

Letting $T_{\text{com}}\in\mathcal{R}^{++}$ and $\tilde{T}_{\text{com}}=N\cdot T_{\text{com}}$, we write the clock phase vector as

$$\boldsymbol{\phi}[\tilde{k}]=\tilde{k}\cdot\tilde{T}_{\text{com}}\cdot\mathbf{1}^{(N)}+\boldsymbol{\tau}[\tilde{k}],\tag{9}$$

where $\boldsymbol{\tau}[\tilde{k}]$ denotes the clock phase error vector. Recalling $\left(\boldsymbol{\alpha}_i^{(\chi)}\right)^T\mathbf{1}^{(N)}=0$ it follows that $\mathsf{A}^{(\chi)}\cdot\mathbf{1}^{(N)}=\mathbf{0}^{(N)}$, where $\mathbf{0}^{(N)}$ is the $N\times 1$ vector of zeros, and we obtain

$$\begin{aligned}&\tilde{k}\cdot\tilde{T}_{\text{com}}\cdot\mathbf{1}^{(N)}+\boldsymbol{\tau}[\tilde{k}]\\&=\begin{cases}\tilde{k}\cdot\tilde{T}_{\text{com}}\cdot\mathbf{1}^{(N)}+2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2], & ((\tilde{k}))_3=0\\\tilde{k}\cdot\tilde{T}_{\text{com}}\cdot\mathbf{1}^{(N)}+2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2]\\+\varepsilon_\phi\mathsf{A}^{(\phi)}\boldsymbol{\tau}[\tilde{k}-1]+\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}, & ((\tilde{k}))_3=1,\\\tilde{k}\cdot\tilde{T}_{\text{com}}\cdot\mathbf{1}^{(N)}+2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2]\\-\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}-\varepsilon_\phi\mathsf{A}^{(\phi)}\boldsymbol{\tau}[\tilde{k}-2]\\\quad+\varepsilon_T\mathsf{A}^{(T)}\left(\boldsymbol{\tau}[\tilde{k}-3]-\boldsymbol{\tau}[\tilde{k}-4]\right), & ((\tilde{k}))_3=2.\end{cases}\end{aligned}$$

Simplifying this expression we finally arrive at

$$\boldsymbol{\tau}[\tilde{k}]=\begin{cases}2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2], & ((\tilde{k}))_3=0\\2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2]\\+\varepsilon_\phi\mathsf{A}^{(\phi)}\boldsymbol{\tau}[\tilde{k}-1]+\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}, & ((\tilde{k}))_3=1,\\2\boldsymbol{\tau}[\tilde{k}-1]-\boldsymbol{\tau}[\tilde{k}-2]\\-\varepsilon_\phi\boldsymbol{\gamma}^{(\phi)}-\varepsilon_\phi\mathsf{A}^{(\phi)}\boldsymbol{\tau}[\tilde{k}-2]\\\quad+\varepsilon_T\mathsf{A}^{(T)}\left(\boldsymbol{\tau}[\tilde{k}-3]-\boldsymbol{\tau}[\tilde{k}-4]\right), & ((\tilde{k}))_3=2.\end{cases}\tag{10}$$

Observe that $\tilde{T}_{\text{com}}$ does not affect convergence. The resulting system (10) is a vector linear, periodically time-varying system, whose dimension is $N\times 1$ and its period is 3. This system can be equivalently represented as a $3N\times 1$ linear, time-invariant system as follows:

$$\begin{bmatrix} \mathsf{I}_N & -2\mathsf{I}_N & \mathsf{I}_N \\ \mathsf{0}_N & \mathsf{I}_N & -2\mathsf{I}_N \\ \mathsf{0}_N & \mathsf{0}_N & \mathsf{I}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}[\tilde{\tilde{k}}] \\ \boldsymbol{\tau}[\tilde{\tilde{k}}-1] \\ \boldsymbol{\tau}[\tilde{\tilde{k}}-2] \end{bmatrix}$$

$$= \begin{bmatrix} \mathsf{0}_N & \mathsf{0}_N & \mathsf{0}_N \\ -\mathsf{I}_N - \varepsilon_\phi \mathsf{A}^{(\phi)} & \varepsilon_T \mathsf{A}^{(T)} & -\varepsilon_T \mathsf{A}^{(T)} \\ 2\mathsf{I}_N + \varepsilon_\phi \mathsf{A}^{(\phi)} & -\mathsf{I}_N & \mathsf{0}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}[\tilde{\tilde{k}}-3] \\ \boldsymbol{\tau}[\tilde{\tilde{k}}-4] \\ \boldsymbol{\tau}[\tilde{\tilde{k}}-5] \end{bmatrix} + \begin{bmatrix} \mathbf{0}_N \\ -\varepsilon_\phi \boldsymbol{\gamma}^{(\phi)} \\ \varepsilon_\phi \boldsymbol{\gamma}^{(\phi)} \end{bmatrix} \tag{11}$$

$$\Leftrightarrow \ \mathsf{B} \cdot \mathbf{y}[\tilde{m}] = \mathsf{C} \cdot \mathbf{y}[\tilde{m}-1] + \mathbf{u}$$
$$\Leftrightarrow \ \mathbf{y}[\tilde{m}] = \mathsf{B}^{-1} \cdot \mathsf{C} \cdot \mathbf{y}[\tilde{m}-1] + \mathsf{B}^{-1} \cdot \mathbf{u} \tag{12}$$

where $\tilde{\tilde{k}} = 3 \cdot \tilde{m} \cdot \tilde{k}$ for $\tilde{m} \in \mathcal{Z}^{++}$, and B, C, $\mathbf{y}[\tilde{m}]$, and $\mathbf{u}$ in Eqn. (12) are defined as the respective quantities in Eqn. (11).

Using elementary matrix operations we can explicitly compute

$$\mathsf{B}^{-1}\mathsf{C} = \begin{bmatrix} 4\mathsf{I}_N + \varepsilon_\phi \mathsf{A}^{(\phi)} & 2\varepsilon_T \mathsf{A}^{(T)} - 3\mathsf{I}_N & -2\varepsilon_T \mathsf{A}^{(T)} \\ 3\mathsf{I}_N + \varepsilon_\phi \mathsf{A}^{(\phi)} & \varepsilon_T \mathsf{A}^{(T)} - 2\mathsf{I}_N & -\varepsilon_T \mathsf{A}^{(T)} \\ 2\mathsf{I}_N + \varepsilon_\phi \mathsf{A}^{(\phi)} & -\mathsf{I}_N & 0 \end{bmatrix}.$$

Clearly, when the network is strongly connected then the system matrix is irreducible, which is a necessary condition for facilitating network clock synchronization [12, Thm. 1]. We also observe that $\mathsf{A}^{(\phi)} \cdot \mathbf{1}_N = \mathsf{A}^{(T)} \cdot \mathbf{1}_N = \mathbf{0}_N$, which is also a necessary condition for the existence of a steady state. Note that the case of a second order loop is considered difficult to solve analytically [12]; hence we evaluated the eigenvalues of $\mathsf{B}^{-1}\mathsf{C}$ for 800 realizations in the numerical evaluation in Sec. VI. It was observed that in all deployments no eigenvalues were outside the unit circle. Applying the vector Z-transform to Eqn. (12) we obtain

$$\mathsf{B} \cdot \mathbf{Y}(z) = \mathsf{C} \cdot \mathbf{Y}(z) \cdot z^{-1} + \mathbf{u}\left(1 - z^{-1}\right)^{-1}$$
$$\Rightarrow \quad \left(\mathsf{B} - \mathsf{C}z^{-1}\right) \cdot \mathbf{Y}(z) = \mathbf{u} \cdot \left(1 - z^{-1}\right)^{-1}$$
$$\mathbf{Y}(z) = \left(\mathsf{B} - \mathsf{C}z^{-1}\right)^{-1} \mathbf{u} \cdot \left(1 - z^{-1}\right)^{-1}.$$

Lastly, we apply the final value theorem to obtain the asymptotic value of $\mathbf{y}[\tilde{m}]$:

$$\begin{aligned} \mathbf{y}_\infty \equiv \lim_{\tilde{m} \to \infty} \mathbf{y}[\tilde{m}] &= \lim_{z \to 1} (z-1)\mathbf{Y}(z) \\ &= \lim_{z \to 1} (z-1)\left(\mathsf{B} - \mathsf{C}z^{-1}\right)^{-1} \mathbf{u} \frac{z}{z-1} \\ &= \left(\mathsf{B} - \mathsf{C}\right)^{-1} \mathbf{u} \end{aligned} \tag{13}$$

Note that we are interested in the asymptotic NPDR which is expressed as

$$\mathrm{NPDR}[\tilde{k}] \underset{\tilde{k} \to \infty}{\longrightarrow} \frac{\max\{\mathbf{y}_\infty\} - \min\{\mathbf{y}_\infty\}}{T_{\mathrm{com}}} \tag{14}$$

where T_{com} is the common clock period in (9).

B. Performance Evaluation of Distributed Synchronization With Previously Proposed Weights

1) Nested Loops With Weights Computed According to Relative Power: In [5] and [12], the weights were computed according to the relative power. Explicitly, for $\chi \in \{\phi, T\}$, the weights $\alpha_{i,j}^{(\chi)}$ are set to [5, Eqn. (8)]

$$\alpha_{i,j}^{(\chi)} = \alpha_{i,j} = \frac{P^{(i)}[j]}{\sum\limits_{m \in \mathcal{N}(i)} P^{(i)}[m]}, \tag{15}$$

It is verified that $\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} = 1$, hence, setting $\alpha_{i,i} = 0$ we can also write

$$\sum_{j=1}^{N} \alpha_{i,j} = 1, \qquad i = 1, 2, \ldots, N. \tag{16}$$

We refer to the nested phase and frequency synchronization algorithm (NPFSA) with the weights of (15) as the relative power algorithm (RPA).

2) Nested Loops With Equal Weights: A simple alternative to the RPA is weighing all period and phase errors equally across the received nodes, i.e.,

$$\alpha_{i,j}^{(\phi)} = \alpha_{i,j}^{(T)} = \alpha_{i,j} = 1/|\mathcal{N}(i)| \tag{17}$$

Again, setting $\alpha_{i,i} = 0$, (16) is satisfied with the assignment (17). We refer to the NPFSA with weights (17) as the equal-weights algorithm (EWA). Such an approach was analyzed in [27], where the optimal bandwidth was derived, it was also shown that local computation at the nodes is globally optimal (subject to using equal weights).

The performance of the EWA and RPA for the *representative scenario* of Fig. 2 is depicted in Fig. 8 for $\varepsilon_T = \varepsilon_\phi = 0.3$. Figs. 8a and 8b depict the evolution of the clock periods. It can be observed that the RPA requires a very long time to accurately synchronize the periods, while the EWA quickly achieved high accuracy. This can be attributed to the fact that for the RPA, the system matrix $\mathsf{B}^{-1}\mathsf{C}$ has more eigenvalues close to the unit circle than for the EWA. Fig. 8c depicts the NPDR for both algorithms. It is evident from the figure that the

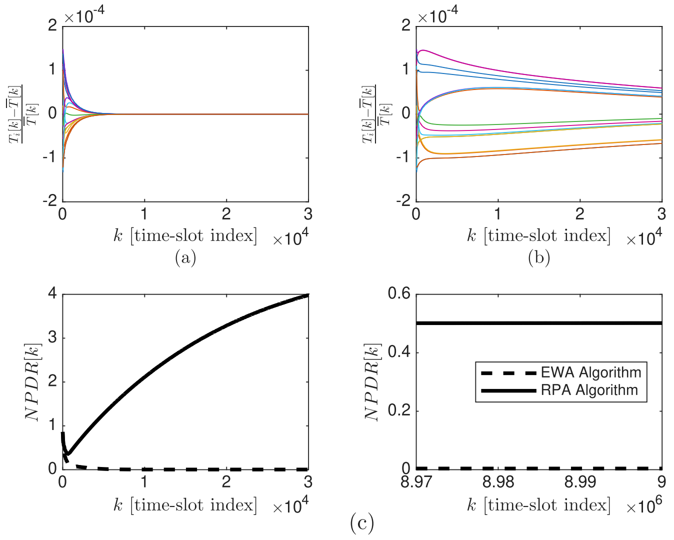


Fig. 8. Performance of the EWA and the RPA with $\varepsilon_T = \varepsilon_\phi = 0.3$ for the *representative scenario.* (a) Normalized clock periods $T_{\mathrm{com}} + T_i[k]$ vs. k for the EWA; (b) Normalized clock periods $T_{\mathrm{com}} + T_i[k]$ vs. k for the RPA. In (a) and (b) each line represents the trajectory of the clock period at a different node. (c) The NPDR vs. k for both the EWA and the RPA.

TABLE I
MEAN AND STD OF THE ASYMPTOTIC NPDR FOR THE EWA AND RPA

	$\mathbb{E}$(NPDR[k]) measured	STD(NPDR[k]) measured	$\mathbb{E}$(NPDR[k]) analytic	STD(NPDR[k]) analytic
EWA representative	0.0040	–	0.0040	–
RPA representative	0.5013	–	0.5938	–
EWA statistical	0.0043	0.0024	0.0043	0.0024
RPA statistical	0.2457	0.2039	0.2442	0.1923

RPA is not able to achieve accurate clock synchronization, and in fact, as time increases the clocks' phases drift farther apart, eventually converging at a very high NPDR. It is also evident that the EWA achieves superior convergence performance compared to the RPA. Finally, note that the EWA's weighting scheme does not require measurement of the received signal power. This suggests the suboptimality of the intuitive weights of (15).

We next compare the asymptotic NPDR according to the analytical expression with the measured results, for both the EWA and the RPA, both for the representative network depicted in Fig. 2 and a statistical comparison with 100 network realizations. The results are summarized in Table I. Observe that there is a very good match between the measured results and the analytic results. The difference for the RPA is attributed to the extremely slow convergence due to poles closer to the unit circle. Note that for the *representative scenario* a single realization is tested thus the standard deviation (STD) is irrelevant and the mean is equal to the final NPDR value.

C. Discussion

From the above analysis we draw several important insights:

- First, we note that the novelty and rationale of the proposed decoupled structure are validated by the analysis reported above: Without decoupling the phase and period updates, it is analytically shown in the Appendix that the frequencies of the clocks do not reach perfect alignment, namely, there remain clock period differences after convergence. In our analysis of the decoupled scheme we show that as no eigenvalues of the system matrix $\mathsf{B}^{-1}\mathsf{C}$ are outside the unit circle, then the phase differences between the nodes reach constant values which imply that the periods *reach the same value*. Thus, the decoupled nested loop structure is able to synchronize the period and phase also when operating in the HD regime, implying that decoupling results in a fundamental improvement in system performance compared to the non-decoupled structure. As far as we can tell, this is the first analysis for a HD synchronization loop, and the first one to derive performance for decoupled updates. The derivation shows that the decoupled algorithm is represented by a linear, periodically time-varying system.
- We observe, that *if the propagation delays are neglected*, the nested loop is able to synchronize both phase and frequency with (nearly) zero error, even if the initial clocks' periods are different. This is in contrast to the situation with only a phase loop, where it was shown that different clock frequencies induce asymptotic phase mismatch [5]. It follows that the *inherent factor which limits synchronization accuracy is the propagation delays*, whereas different clock frequencies can be algorithmically addressed.
- The analysis shows that the weights (i.e., the matrices $\mathsf{A}^{(\phi)}$, $\mathsf{A}^{(T)}$) not only affect the *rate of convergence* but also affect the *asymptotic NPDR*.
- If the propagation delays are known (e.g., fixed locations) then (14) can be used to determine the optimal weights, in the sense of minimal NPDR.
- As T_{com} does not affect convergence, it can be selected arbitrarily. One possible option is to set a reference node that will not adjust its clock period. The analysis above implies that all nodes will perfectly synchronize their periods with the reference node.

The facts that the optimal weights depend on parameters which are inherently a-priori unknown and that intuitive assignments do not work well, strongly motivate optimizing the weights after deployment, which is the focus of the next section.

V. DNN-BASED COEFFICIENTS OPTIMIZATION WITH UNSUPERVISED DISTRIBUTED ONLINE LEARNING

In our preliminary work [24] we proposed DNN-based weights that led to superior performance compared to the RPA. Inspired by the good accuracy attained by the EWA, in the current work we significantly improve upon the scheme of [24]: We modify the DNN used in [24] by adding a layer that facilitates initializing some of the DNN parameters s.t. the correction signal generated prior to training approximates the EWA signal. Thus, after training, the DNN-aided algorithm will generally improve upon the EWA and upon [24]. We note that identifying the initial parameters' values that will result in desired DNN output values has largely not been considered previously for this type of DNN applications, and represents another novel aspect of our current work.

A. Overview of the Proposed DNN-Aided Algorithm

Our proposed DNN-aided NPFSA, referred to as the DNN-aided algorithm (DAA), employs learning only for the phase and period correction signals, keeping the other elements of the loop structure of Fig. 4 and the periodic, staggered temporal regime schematically described in Fig. 5, while replacing only the weights $\alpha_{i,j}^{(\phi)}$, $\alpha_{i,j}^{(T)}$ in Eqns. (7), (8), with the outputs of two DNNs. To that aim, let $\psi_{T,i}^{(\boldsymbol{\theta}_{T,i})}(\cdot)$ and $\psi_{\phi,i}^{(\boldsymbol{\theta}_{\phi,i})}(\cdot)$ denote the DNNs used for computing the weights for the period and for the phase corrections at node $i \in \mathcal{I}_N$, respectively, where $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$ denote their respective DNN parameters.

During the "period update" action, which takes place at $((k))_{3N} \in \{2N-1, 3N-2\}$, see Eqn. (7), node i uses weights $\gamma_{i,j}^{(T)}$, generated by the DNN $\psi_{T,i}^{(\boldsymbol{\theta}_{T,i})}(\cdot)$, for computing its period correction signal, while $\Omega_i[k] = 0$. The weights $\gamma_{i,j}^{(T)}$ are computed at node *i* at time interval *k* s.t. $((k))_{3N} = 2N-1$, via

$$\gamma_{i,j}^{(T)} = \left[\psi_{T,i}^{(\boldsymbol{\theta}_{T,i})}\left(\left\{\left(\Delta_T^{(i)}[j'], P^{(i)}[j']\right)\right\}_{\substack{j'=1,\\ j'\neq i}}^{N}\right)\right]_j, \quad \gamma_{i,i}^{(T)} = 0. \tag{18}$$

The period correction signal at node i is given by

$$\Delta T_i[k]=\begin{cases}\frac{\varepsilon_T}{N}\sum_{j=1}^{N}\gamma_{i,j}^{(T)}\cdot\Delta_T^{(i)}[j]\,, & ((k))_{3N}=2N-1\\ \Delta T_i[k-1], & 2N\leq((k))_{3N}\leq 3N-2\\ 0, & \text{otherwise.}\end{cases} \tag{19}$$

Subsequently, during the "phase update" action, when k is s.t. $((k))_{3N}=3N-1$, an update is applied by weighting the received clock phase differences with weights $\gamma_{i,j}^{(\phi)}$ generated by the DNN $\psi_{\phi,i}^{(\boldsymbol{\theta}_{\phi,i})}(\cdot)$, while $\Delta T_i[k]=0$. The weights $\gamma_{i,j}^{(\phi)}$ are then computed at node i via

$$\gamma_{i,j}^{(\phi)}=\left[\psi_{\phi,i}^{(\boldsymbol{\theta}_{\phi,i})}\left(\left\{\left(\Delta_\phi^{(i)}[j'],P^{(i)}[j']\right)\right\}_{\substack{j'=1,\\ j'\neq i}}^{N}\right)\right]_j,\ \gamma_{i,i}^{(\phi)}=0, \tag{20}$$

and the phase correction signal at node i is given by

$$\Omega_i[k]=\begin{cases}\varepsilon_\phi\sum_{j=1}^{N}\gamma_{i,j}^{(\phi)}\cdot\Delta_\phi^{(i)}[j]\,, & ((k))_{3N}=3N-1\\ 0, & \text{otherwise.}\end{cases} \tag{21}$$

The overall algorithm operates as in Fig. 4, with the $\alpha_{i,j}^{(T)}$'s in (7) replaced with $\gamma_{i,j}^{(T)}$ given in (18), and $\alpha_{i,j}^{(\phi)}$'s in (8) replaced with $\gamma_{i,j}^{(\phi)}$ given in (20).

B. Operation of the Proposed DAA (at Node i)

In [24], the DNNs were trained right after startup with random initial parameters $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$. It is reasonable to assume that further improvement in performance could be achieved if training would start when the clocks have already achieved a certain degree of synchronization *prior* to training data acquisition. The difficulty here lies as it is generally unknown how to set the initial values of $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$ to obtain desired weights for initial convergence. In this work we propose a DNN structure with an initialization which facilitates an approximate EWA prior to training. To that aim we add a layer before the last layer in the DNN of [24], designated *as a "bias layer"*, which adds a predetermined bias value to each input. Note that during training these biases are optimized as well. The outputs of the "bias layer" are then scaled by the last layer s.t. they add up to 1. Let $x_l\in[0,1]$ denote the l-th input to the "bias layer", *where the x_l's are scaled s.t.* $\sum_{l\in\mathcal{N}(i)}x_l=1$. Letting b_l denote the bias applied to the l-th input, the l-th output of the bias layer is expressed as x_l+b_l. After scaling such that the sum of the outputs will be 1, we obtain the l-th weight as

$$\begin{aligned}w_l&=\frac{x_l+b_l}{\sum_{l'\in\mathcal{N}(i)}(x_{l'}+b_{l'})}\\&=\frac{x_l+b_l}{\sum_{l'\in\mathcal{N}(i)}b_{l'}+1}\in\left[\frac{b_l}{1+\sum_{l'\in\mathcal{N}(i)}b_{l'}},\frac{b_l+1}{1+\sum_{l'\in\mathcal{N}(i)}b_{l'}}\right].\end{aligned}$$

It thus follows that the initial value of b_l can be selected such that the weights w_l are approximately equal.

The proposed DAA operates as follows: At startup, the DNN parameters $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$ are randomly selected, except for the parameters of the "bias layer" b_l, $l\in\mathcal{N}(i)$ which are set to a constant $b_l=B$, and b_l, $l\notin\mathcal{N}(i)$ which are set to zero. The network then begins operating for an initial interval of N_{init} time-slot intervals, while the parameters $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$ are kept fixed. This operation *approximates the EWA*, hence given sufficient time it would typically achieve a relatively accurate synchronization. After N_{init} samples, the DAA begins data acquisition over N_T TDMA frames, or, equivalently over $N\cdot N_T$ time-slot intervals. Subsequently, the DAA applies local training to the parameter vectors $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$ at each node for a total of $2\cdot E_{\text{loop}}\cdot E_s$ epochs, after which the DAA continues to operate with the trained DNNs. In Sec. V-D we elaborate on the acquisition and training steps.

C. DNN Structure and Complexity Analysis

The frequency synchronization loop and the phase synchronization loop use the same DNN structure, depicted in Fig. 9. Each DNN consists of a total of ten layers, including two affine layers, each followed by a sigmoid layer, a third linear layer followed by a softmax layer, whose outputs are input to an affine layer referred to as the *bias layer* which introduces a bias as detailed in Sec. V-B. A mask is then applied to the output of the bias layer, setting all outputs that correspond to network nodes not received at the current node to zero. After

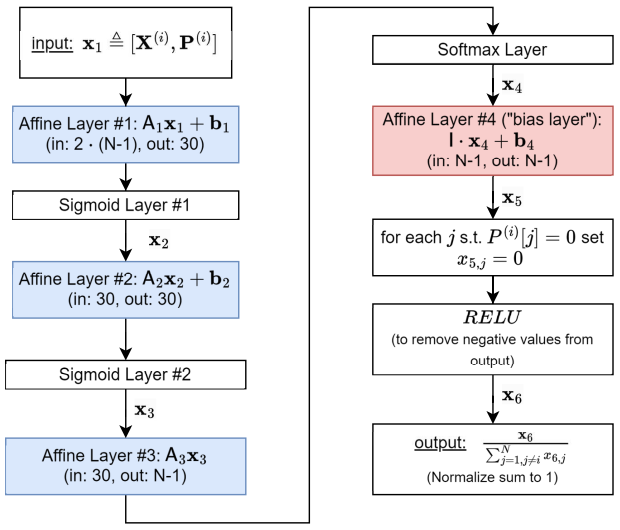

input: $\mathbf{x}_1\triangleq[\mathbf{X}^{(i)},\mathbf{P}^{(i)}]$
Affine Layer #1: $\mathbf{A}_1\mathbf{x}_1+\mathbf{b}_1$ (in: 2 · (N-1), out: 30)
Sigmoid Layer #1
$\mathbf{x}_2$
Affine Layer #2: $\mathbf{A}_2\mathbf{x}_2+\mathbf{b}_2$ (in: 30, out: 30)
Sigmoid Layer #2
$\mathbf{x}_3$
Affine Layer #3: $\mathbf{A}_3\mathbf{x}_3$ (in: 30, out: N-1)
Softmax Layer
$\mathbf{x}_4$
Affine Layer #4 ("bias layer"): $\mathbf{I}\cdot\mathbf{x}_4+\mathbf{b}_4$ (in: N-1, out: N-1)
$\mathbf{x}_5$
for each j s.t. $P^{(i)}[j]=0$ set $x_{5,j}=0$
RELU (to remove negative values from output)
$\mathbf{x}_6$
output: $\frac{\mathbf{x}_6}{\sum_{j=1,j\neq i}^{N}x_{6,j}}$ (Normalize sum to 1)

Fig. 9. Schematic description of the DNN. $\mathbf{X}^{(i)}\equiv\{\Delta_\phi^{(i)}[j]\}_{j\in\mathcal{N}(i)}$ for phase correction, and $\mathbf{X}^{(i)}\equiv\{\Delta_T^{(i)}[j]\}_{j\in\mathcal{N}(i)}$ for period correction.

TABLE II
SUMMARY OF THE LAYERS IN THE DNN IMPLEMENTATION

Operation	Inputs	Outputs
Affine Layer 1	$2(N-1)$	30
Sigmoid Layer 1	30	30
Affine Layer 2	30	30
Sigmoid Layer 2	30	30
Affine Layer 3	30	$N-1$
Softmax Layer	$N-1$	$N-1$
Affine Layer 4	$N-1$	$N-1$
Masking	$N-1$	$N-1$
RELU	$N-1$	$N-1$
Normalization	$N-1$	$N-1$

masking, all negative outputs are set to zero and the remaining outputs are scaled such that they add up to one. These outputs are the weights applied in the loop. The number of inputs and outputs of each layer are summarized in Table II.

For a network with N nodes, each DNN consists of $(3(N-1)+30)\cdot 30$ weights and $2\cdot 30+(N-1)$ biases. For a network with $N=16$ nodes, we obtain 2250 weights and 75 biases per DNN. We note that, since each DNN is used for inference only once per $3\cdot N$ time slots, and there are two DNNs at each node, the rate of computation at a node is 1500 products for inference per TDMA frame, which is a feasible computational load for real-time modern microcontrollers, see [11] and [28]. We note that apart from the computation of the weights, the three synchronization algorithms discussed in Section IV-B and in this section, namely, the EWA, RPA, and DAA have the same complexity. Hence, the complexity obtained here represents the excess computational burden of the DAA relative to the RPA and the EWA. Note that this complexity, scales linearly with the number of nodes N, for $N\leq 16$.

D. Unsupervised Distributed Online Training

Typically, DNNs are trained offline with data collected a-priori. However, as follows from the analysis in Sec. IV and was also empirically observed in [11], the accuracy of DNN-aided clock synchronization can be improved by using training data that corresponds to the actual parameters of the network deployment, i.e., the actual clock frequency differences, clock phases, and, most critically, propagation delays and neighborhood sets. This requires collecting the training data after deployment. Accordingly, the training process consists of three steps:

Step 1: A free running step where each node applies its clock update algorithm with fixed DNN parameters, judiciously selected to approximate the EWA. We note that by the analysis in [12], it follows that the EWA is convergent for a first order phase loop. From the extensive simulation tests reported in Sec. VI it follows that the EWA is expected to converge for the current setup, as discussed in Secs. IV-A and IV-C.

Step 2: A data acquisition step is applied following the free running step. In this step, each node stores the time stamps for the clock signatures received from its neighboring nodes along with power measurements of the signature signals.

Step 3: Finally, in the third step, training is applied, minimizing the loss functions. As explained in Sec. V-D.3, the structure of the proposed loss functions facilitates using back-propagation to compute the gradients.

We note that throughout this process, the clock synchronization algorithm at each node continues its clock update operation with its initial DNN parameters. The timeline of the different steps of the algorithm is depicted in Fig. 10. In the following subsections we elaborate on the steps of the training algorithm.

1) Initialization of DNNs' Parameters $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$: As detailed in Sec. V-B we initialize the "bias layer" parameters represented by the vector $\mathbf{b}_4$ in "Affine Layer 4" in Fig. 9, to

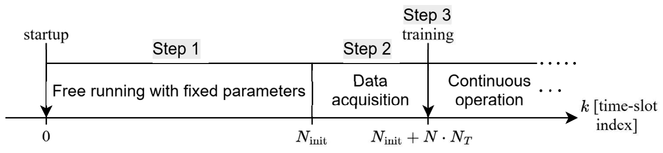


Fig. 10. Timeline for the three steps of the DAA training process.

values B while the remaining DNN parameters in $\boldsymbol{\theta}_{T,i}$ and in $\boldsymbol{\theta}_{\phi,i}$ are generated randomly. The synchronization scheme then operates with fixed parameters for N_{init} time-slot intervals, after which the data acquisition and training step is applied, as described next.

2) Data Acquisition and Training Set Generation: The data acquisition step begins at $k=N_{\text{init}}+1$, and spans N_T TDMA frames, each consists of N transmissions, one from each node. During data acquisition, each node i continues updating its clock times via the DAA with its fixed initial parameters, $\boldsymbol{\theta}_{\phi,i}$ and $\boldsymbol{\theta}_{T,i}$, determined as detailed in Sec. V-D.1. At time-slot interval k, node $j=((k))_N+1$ transmits, and each node $i\in\mathcal{I}_N\setminus j$ receives. Thus, at every time k, $N_{\text{init}}+1\leq k\leq N_{\text{init}}+N\cdot N_T$, node i stores the pair $\{\Delta t_{i,j}[k]+\phi_i[k], P_{i,j}[k]\}$, where $j=((k))_N+1$, $j\neq i$. After receiving N_T TDMA frames, each node i generates a training set containing N_T samples, where each sample consists of $N-1$ pairs. Setting $t_{i,((k))_N+1}[k]\triangleq \Delta t_{i,((k))_N+1}[k]+\phi_i[k]$, the overall training set at node i is given as

$$\mathcal{D}_i=\left\{t_{i,((k))_N+1}[k], P_{i,((k))_N+1}[k]\right\}_{k=N_{\text{init}}+1,((k))_N\neq i-1}^{N_{\text{init}}+N\cdot N_T}$$

3) Training Procedure: Once the acquisition step is completed, node i processes the stored training data at sample k', $N_{\text{init}}+1\leq k'\leq N_{\text{init}}+N\cdot N_T$ as follows: Let $j'=((k'))_N+1$. Then, if $P_{i,j'}[k']>P_{\text{th}}$, the features used for computing the outputs of the DNNs $\psi_{T,i}^{(\boldsymbol{\theta}_{T,i})}(\cdot)$ and $\psi_{\phi,i}^{(\boldsymbol{\theta}_{\phi,i})}(\cdot)$ are obtained in the following order

$$\begin{aligned}
&\Delta_T^{(i)}[j']\leftarrow\left(t_{i,j'}[k']-\phi_i[k']-\Delta_\phi^{(i)}[j']\right)/N\\
&\Delta_\phi^{(i)}[j']\leftarrow t_{i,j'}[k']-\phi_i[k']\\
&P^{(i)}[j']\leftarrow P_{i,j'}[k'].
\end{aligned}$$

If $P_{i,j'}[k']\leq P_{\text{th}}$, node i sets $\Delta_T^{(i)}[j']=\Delta_\phi^{(i)}[j']=P^{(i)}[j']=0$. Subsequently, the DNNs compute the weights for the phase and for the period loops at node i, using the respective $2(N-1)$ features at each DNN according to the time regime via Eqns. (5), (18), (19), (20), and (21). Let

$$\Delta T_{i,j'}[k']=\frac{1}{N}\left(t_{i,j'}[k']-\phi_i[k']-t_{i,j'}[k'-N]+\phi_i[k'-N]\right). \tag{22}$$

Note that $\Delta T_{i,j'}[k']$ represents the estimate of the difference between the clock periods at nodes i and j' as measured at node i at time index k'. Next, let $\mathbb{1}_{(i,j')}(k')$ denote the indicator function for the event $\left\{\left\{P_{i,j'}[k']>P_{\text{th}}\right\}\cap\left\{i\neq j'\right\}\right\}$.

The loss functions are computed as follows:

$$\mathcal{L}_i^{(\phi)}(\boldsymbol{\theta}_{i,\phi}) = \sum_{\substack{k'=N_{\text{init}}+N+1 \\ j'=((k'))_N+1}}^{N_{\text{init}}+N_T \cdot N} \mathbb{1}_{(i,j')}(k') \log(k''(k')) \cdot (t_{i,j'}[k'] - \phi_i[k'])^2 \tag{23}$$

$$\mathcal{L}_i^{(T)}(\boldsymbol{\theta}_{i,T}) = \sum_{\substack{k'=N_{\text{init}}+N+1 \\ j'=((k'))_N+1}}^{N_{\text{init}}+N_T \cdot N} \mathbb{1}_{(i,j')}(k') \log(k''(k')) \cdot \left(\Delta T_{i,j'}[k']\right)^2, \tag{24}$$

where $k''(k') = k' - N_{\text{init}} - N$.

The structure of the loss functions in Eqns. (23), (24), implies that their gradients w.r.t. the weights can be computed via back-propagation in time. Furthermore, these loss functions can be computed at each node locally in an unsupervised manner. This training process is summarized in Alg. 1, which expands upon [24, Alg. 1] to handle the new initialization. Observe that at each training batch, first, only the period loop is trained, and then only the phase loop is trained. This is done to avoid the phase loop compensating for period errors, which would decrease the accuracy of period synchronization across the nodes. Finally, note that as training is local at each node and uses only its received inputs obtained during the data acquisition step, the proposed approach is flexible and would generally scale with the size of the network (as long as the number of inputs at the first layer is larger than $2(N-1)$ and the number of outputs at the third layer and at the subsequent layers is larger than $N-1$). The overall operation of the DAA is summarized in Alg. 2.

Comment 1 (Regarding Convergence of the DAA): There are quite a few works on the convergence of distributed learning using stochastic gradient descent, assuming the loss is strongly convex w.r.t. the weights, e.g., [29], [30]. Note that the sigmoid function used in the DNN structure depicted in Fig. 9 is not convex, which makes showing convergence analytically a very challenging task. This issue is not special to our study, and is a universal issue with the application of machine learning in distributed settings. That said, we note that the weights obtained from the trained DNNs satisfy the two fundamental properties of the analytical weights: They are positive and add to 1. Thus, based on the analysis for the RPA and the EWA reported in Sec. IV, the proposed DAA satisfies necessary conditions for convergence. Furthermore, as optimization begins with weights that approximate the EWA, it would generally result in performance improvement over the EWA. In Sec. VI we empirically tested 800 random realizations of strongly connected networks, and observed that convergence was achieved for all tested network realizations. All these observations indicate that the proposed DAA is very likely to converge.

Algorithm 1 Unsupervised Online Local Training at Node i

```
Data: Data set $\mathcal{D}_i$ consisting of $N_T$ tuples, each of length $3 \cdot (N-1)$; learning rate $\mu$; initial parameters ($\boldsymbol{\theta}_{\phi,i}$, $\boldsymbol{\theta}_{T,i}$), number of epochs $E_s$ and number of epochs for each parameter $E_{\text{loop}}$, clock phases and periods for the first $N$ indices after $N_{\text{init}}$, $\{\phi_i[k'], T_i[k']\}_{k'=N_{\text{init}}+1}^{N_{\text{init}}+N+1}$.
for epoch $= 1$ to $E_s$ do
    // Initialize:
    forall $k' \in \{N_{\text{init}}+1, N_{\text{init}}+2, \ldots, N_{\text{init}}+N\}$ do
        Set current Tx node $j' = ((k'))_N + 1$
        if $j' \neq i$ then
            Read $\{t_{i,j'}[k'], P_{i,j'}[k']\} \in \mathcal{D}_i$
            if $P_{i,j'}[k'] > P_{th}$ then
                Set $\Delta_\phi^{(i)}[j'] = t_{i,j'}[k'] - \phi_i[k']$
            else
                Set $\Delta_\phi^{(i)}[j'] = 0$
            end
        end
    end
    // Compute Updates:
    for $m = 1$ to $2 \cdot E_{loop}$ do
        for $k' = N_{\text{init}} + N + 1$ to $N_{\text{init}} + N \cdot N_T$ do
            Set current Tx node $j' = ((k'))_N + 1$
            if $j' \neq i$ then
                Read $\{t_{i,j'}[k'], P_{i,j'}[k']\} \in \mathcal{D}_i$
                if $P_{i,j'}[k'] > P_{th}$ then
                    Add $j'$ to $\mathcal{N}(i)$
                    Compute $\Delta\phi_{i,j'} = t_{i,j'}[k'] - \phi_i[k']$
                    Store $\Delta_T^{(i)}[j'] \leftarrow (\Delta\phi_{i,j'} - \Delta_\phi^{(i)}[j'])/N$
                    Store $\Delta_\phi^{(i)}[j'] \leftarrow \Delta\phi_{i,j'}$
                    Store $P^{(i)}[j'] \leftarrow P_{i,j'}[k']$
                else
                    Exclude $j$ from $\mathcal{N}(i)$
                    Store
                     $\Delta_T^{(i)}[j'] = \Delta_\phi^{(i)}[j'] = P^{(i)}[j'] = 0$
                end
            end
            Forward pass $P^{(i)}[j']$, $\Delta_T^{(i)}[j']$ and $\Delta_\phi^{(i)}[j']$ to
             compute $\phi_i[k'+1]$ and $T_i[k'+1]$ via Eqns.
             (5), (18)–(21)
        end
        // Compute Loss and Update Weights:
        if $m \leq E_{loop}$ then
            Compute loss $\mathcal{L}_i(\boldsymbol{\theta}_{i,T})$ via Eqns. (24) and (22),
             for $k' \in \{N_{\text{init}} + N + 1, \ldots, N_{\text{init}} + N \cdot N_T\}$
            Compute gradient $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i(\boldsymbol{\theta}_{i,T})$ using
             backpropagation through time
            Update weights via
             $\boldsymbol{\theta}_{i,T} \leftarrow \boldsymbol{\theta}_{i,T} - \mu \cdot \nabla_{\boldsymbol{\theta}_{i,T}}\mathcal{L}_i(\boldsymbol{\theta}_{i,T})$
        else
            Compute loss $\mathcal{L}_i(\boldsymbol{\theta}_{i,\phi})$ via Eqn. (23) using the
             computed $\phi_i[k']$ and the stored $t_{i,j}[k']$, for
             $k' \in \{N_{\text{init}} + N + 1, \ldots, N_{\text{init}} + N \cdot N_T\}$
            Compute gradient $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_i(\boldsymbol{\theta}_{i,\phi})$ using
             backpropagation through time
            Update weights via
             $\boldsymbol{\theta}_{i,\phi} \leftarrow \boldsymbol{\theta}_{i,\phi} - \mu \cdot \nabla_{\boldsymbol{\theta}_{i,\phi}}\mathcal{L}_i(\boldsymbol{\theta}_{i,\phi})$
        end
    end
end
```

VI. Performance Evaluation

A. Baseline Scenario and Reference Schemes

In the simulations we use the network and clock models described in Sec. II. For determining the state-of-the-art, consider first the previously proposed EWA and RPA, reviewed in Sec. IV-B, and note that both the EWA and the RPA have the same nested loop architecture depicted in Fig. 4. An alternative approach to the loop-based architecture is the skew and offset

Algorithm 2 Operation of the DAA at Node i

```
   Data: Number of steps $N_{\text{init}}$; initial bias value $B$;
1  Initialize: Set $\Delta_\phi^{(i)}[j] = \Delta_T^{(i)}[j] = 0$, $\forall j \in \mathcal{I}_N$, $j \neq i$;
     $k = 0$; Randomly generate parameters $\boldsymbol{\theta}_{T,i}$ and $\boldsymbol{\theta}_{\phi,i}$.
     Set the parameters corresponding to $\mathbf{b}_4$ to $B$ in both
     DNNs. Set $\mathcal{D}^{(i)}$ to the empty set
2  repeat
3      Set current Tx node $j = ((k))_N + 1$
4      if $j = i$ then
5          Send signature sequence;
6      else if $((k))_{3N} \in \{0, 1, \ldots, 2N-1\}$ then
7          Receive $\{\Delta t_{i,j}[k], P_{i,j}[k]\}$
8          if $P_{i,j}[k] > P_{th}$ then
9              Add $j$ to $\mathcal{N}(i)$
10             Store $\Delta_T^{(i)}[j] \leftarrow (\Delta t_{i,j}[k] - \Delta_\phi^{(i)}[j])/N$
11             Store $\Delta_\phi^{(i)}[j] \leftarrow \Delta t_{i,j}[k]$
12             Store $P^{(i)}[j] \leftarrow P_{i,j}[k]$
13         else
14             Exclude $j$ from $\mathcal{N}(i)$
15             Store $\Delta_\phi^{(i)}[j] = \Delta_T^{(i)}[j] = P^{(i)}[j] = 0$
16         end
17     end
18     Forward pass $P^{(i)}[j]$, $\Delta_T^{(i)}[j]$ and $\Delta_\phi^{(i)}[j]$ to
         compute $\phi_i[k+1]$ and $T_i[k+1]$ via Eqns. (5),
         (18), (19), (20), and (21)
19     // Store Training Data:
20     if $N_{\text{init}} < k \leq N_{\text{init}} + N \cdot N_T$ and $j \neq i$ then
21         Add $\{\Delta t_{i,j}[k] + \phi_i[k], P_{i,j}[k]\}$ to $\mathcal{D}^{(i)}$
22     end
23     if $k = N_{\text{init}} + N \cdot N_T$ then
24         Apply training of DNN parameters $\boldsymbol{\theta}_{T,i}$ and
             $\boldsymbol{\theta}_{\phi,i}$ via Alg. 1, using $\mathcal{D}^{(i)}$
25     end
26     $k \leftarrow k + 1$
27 until indefinitely;
```

estimation (SOE) method, which was studied in [19] and [31]. Consider the operation of the SOE scheme [19] (after adaptation to the HD regime): Let $T_i^{\text{prd}}[0]$ and $\phi_i[0]$ denote the initial period and initial clock phase. Let $\hat{T}_i^{\text{prd}}[0;k]$ denote the estimate of $T_i^{\text{prd}}[0]$ at time k and $\hat{\phi}_i[0;k]$ denote the estimate of $\phi_i[0]$ at time k. Then, the compensated clock at node i is given by (see [19, Eqn. (3)])

$$\phi_i[k] = \frac{T_i^{\text{prd}}[0]}{\hat{T}_i^{\text{prd}}[0;k]} \cdot k + \phi_i[0] - \hat{\phi}_i[0;k],$$

which ideally should approach some common clock, i.e., $\phi_i[k] \longrightarrow k$. The details of the computation of $\hat{T}_i^{\text{prd}}[0;k]$ and $\hat{\phi}_i[0;k]$ are elaborated in [19].

Our simulation tests in Sec. VI-B show that out of the three considered reference schemes, the SOE, RPA, and EWA, the latter is consistently superior to the former two (in fact, the SOE was not able to converge at all in our simulations). Hence, as follows from Sec. IV-B, we use the EWA as a reference

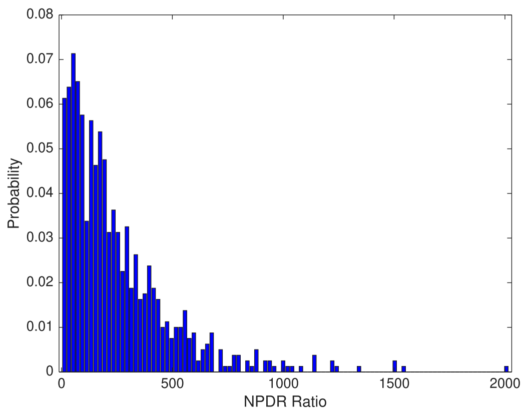


Fig. 11. Histogram of the ratio $\text{NPDR}_{\text{RPA}}[k]/\text{NPDR}_{\text{EWA}}[k]$ at $k = 12000$; $\varepsilon_T = \varepsilon_\phi = 0.3$. Ratios greater than 1 are marked in blue and ratios smaller than 1 are marked in red.

scheme for comparison with the DAA.[3] The training data for the DAA was acquired over $N_T = 126$ TDMA frames, which, for $N = 16$ nodes correspond to 2016 samples, and training was carried out using $E_s = 6$ cycles with $E_{\text{loop}} = 5$. Thus, overall, each DNN was trained over 30 epochs with a learning rate of $\mu = 0.1$. In the testing phase, $N_T = 751$ TDMA frames were used. The bias value for "Affine Layer 4" was set to $B = 3$, and we used $N_{\text{init}} = 3000$ time-slot intervals for initial convergence prior to training data acquisition.

B. Statistical Comparison for Static Deployments

In order to draw meaningful conclusions, we considered multiple random deployments: Statistical performance was obtained by randomly generating 800 network scenarios in which the nodes were randomly and uniformly deployed within the network area, and the initial phases and periods were selected according to the statistics detailed in Sec. II-B. To maintain fairness of comparison, only 800 strongly connected networks scenarios in which about 30% of the links are received above the detection threshold (i.e. *active links*) were considered. We emphasize that in order to characterize performance, the simulation tests must simultaneously account for all the nodes in the network. This follows as the propagation delays are *non-negligible and unknown*. Then, for the *same transmitter's signal*, different receivers record different time stamps due to propagation delays, hence, considering only a pair of nodes would not represent the actual performance.

Fig. 11 depicts the histogram of the NPDR ratio $\text{NPDR}_{\text{RPA}}[k]/\text{NPDR}_{\text{EWA}}[k]$. It is observed that in all the deployments, the EWA achieved smaller NPDR than the RPA, which implies that the EWA should serve as a baseline for evaluating the DAA. Next, Fig. 12 depicts the histogram of the NPDR ratio $\text{NPDR}_{\text{EWA}}[k]/\text{NPDR}_{\text{DAA}}[k]$. It is observed that in 89% of the deployments, the DAA achieved smaller NPDR than the EWA. Note that this is a clear benefit of our initialization approach. It follows that the DAA offers much better performance than the EWA (and hence also better than the ESSBSA).

[3]The complete source code is available at https://github.com/itayzin/Accurate_Clock_Synchronization_HD_TDMA

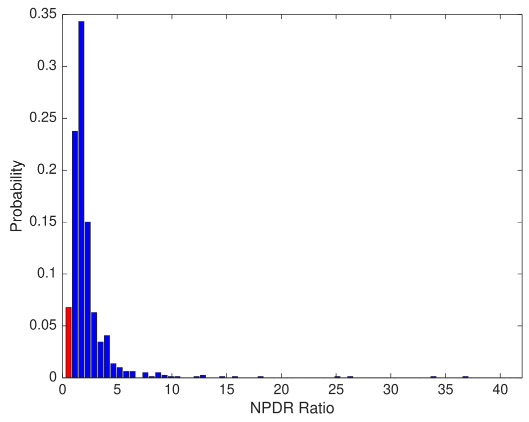


Fig. 12. Histogram of the ratio $\text{NPDR}_{\text{EWA}}[k]/\text{NPDR}_{\text{DAA}}[k]$ at $k = 12000$; $\varepsilon_T = \varepsilon_\phi = 0.3$. Ratios greater than 1 are marked in blue and ratios smaller than 1 are marked in red.

TABLE III

MEAN AND STD OF THE NPDR AND THE PERIOD AT $k = 12000$, FOR 800 EXPERIMENTS.

	$\mathbb{E}$(NPDR[k])	STD(NPDR[k])	$\mathbb{E}\{\bar{T}[k]\}$	STD($\bar{T}[k]$)
EWA	0.010056	0.023607	0.005	$1.1207 \cdot 10^{-7}$
DAA	0.0040202	0.0061011	0.005	$1.1249 \cdot 10^{-7}$
DAARND	0.0066585	0.01719	0.005	$1.2792 \cdot 10^{-7}$

Table III summarizes the values of the mean and the STD of the NPDR at $k = 12000$ for all schemes, together with the mean and STD for the periods. For comparison, we also include the performance of a DAA scheme with all DNN parameters selected randomly (without learning), abbreviated as DAARND. Observe from the table that the mean NPDR of the DAA is significantly smaller than that of the EWA (by a factor of 2.5) and that the STD of the NPDR of the DAA is smaller than that of the EWA by a factor of 3.86. This shows that the DAA achieves a significant improvement in accuracy w.r.t to the reference scheme. It should be noted that the performance of the DAA can be improved by decreasing the step size, but this will also increase the training time. Comparing the DAA with DAARND we observe substantial improvement offered by the proposed parameter initialization: The mean NPDR is reduced by a factor of 1.65 and its STD is reduced by a factor of 2.81.

C. The Impact of Clock Phase Drift on Synchronization

Typically, works that have considered clock synchronization for ad-hoc wireless networks have ignored the impact of clock drift [5], [19], as the clock update rate is typically faster than the dynamics of the drift process. In the following test we verify the validity of this assumption by testing the performance of both the DAA and the EWA in the presence of random clock drift. To this aim we extend the clock model in (1) to include a random drift process, $\xi_i[k]$, $i \in \mathcal{I}_N$, $k \in \mathcal{Z}^+$. Following [32], [33], and [34], we model $\xi_i[k]$ as a first-order autoregressive process (AR (1)), whose steady-state root mean-square (RMS) value corresponds to 1% of the nominal data clock period, denoted T_{data}:

$$\xi_i[k] = \rho \cdot \xi_i[k-1] + \eta_i[k]$$

where $\eta_i[k] \sim \mathbb{N}(0, \sigma^2_{\eta_i})$ is independent across k and is independent of $\xi_i[k-1]$, and the values of $\sigma^2_{\eta_i}$ and ρ are selected s.t. $\sigma_{\eta_i}/\sqrt{1-\rho^2} = 0.01 \cdot T_{\text{data}}$. We set $\rho = 0.999$ as in [35]. The initial noise sample is taken from $\eta_i[0] \sim \mathbb{N}\left(0, \frac{\sigma^2_{\eta_i}}{1-\rho^2}\right)$. The overall clock model used in this test is thus

$$\phi_i[k+1] = \phi_i[k] + T_{\text{com}} + T_i[k] + \xi_i[k], \qquad k \in \mathcal{Z}^+.$$

To determine T_{data} we consider a bit rate of 800 [Kbps] with 4 bits per symbol, hence $T_{\text{data}} = 4/8 \cdot 10^{-5} = 5$ [μsec]. The steady state RMS drift value is then $0.5 \cdot 10^{-7}$ [sec].

To accommodate the correlation of the drift process in the DAA training, we increase the number of training data to 1250 TDMA frames. Fig. 13 depicts a sample path of the NPDR values for the representative deployment of Fig. 2. The random variations of the NPDR due to the random drift are evident. Also, observe the visible improvement in the NPDR due to training which concludes at $k = 24000$. Next, Fig. 14 presents the histogram of the NPDR ratio $\text{NPDR}_{\text{EWA}}[k]/\text{NPDR}_{\text{DAA}}[k]$ in the presence of random clock drift, taken with 800 random realizations. Noting that the NPDR is a random process, we use the average of the NPDRs over the last 150 TDMA frames, ending at time-slot $k = 100000$, as the NPDR values for the comparative histogram. From the histogram we obtain that the mean value and the STD of the NPDR ratio, $\text{NPDR}_{\text{EWA}}[k]/\text{NPDR}_{\text{DAA}}[k]$, are 1.55 and 0.59 respectively. We also evaluated the normalized STD for period estimation $\text{STD}(\bar{T}[k])/\mathbb{E}\{\bar{T}[k]\}$, obtaining $0.83 \cdot 10^{-3}$ for the EWA and $0.89 \cdot 10^{-3}$ for the DAA. With DAA being superior at over 80% of the realizations, it follows that the clear advantage of the DAA over the EWA is maintained also under random drift, as expected.

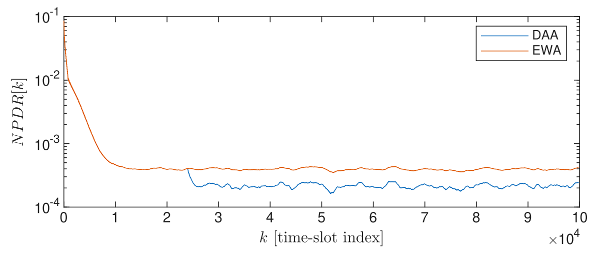


Fig. 13. Sample path for $\text{NPDR}_{\text{EWA}}[k]$ (brown) and $\text{NPDR}_{\text{DAA}}[k]$ (blue) vs. slot index k, before and after training in the presence of random clock drift; $\varepsilon_T = \varepsilon_\phi = 0.3$.

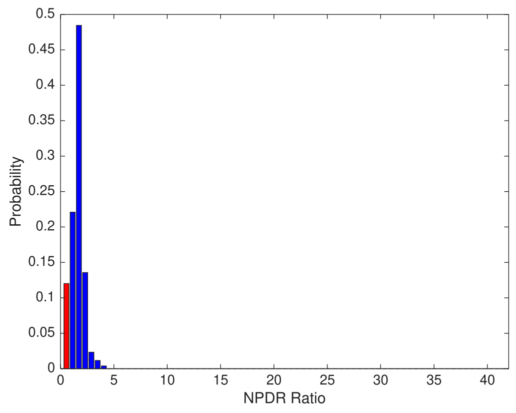


Fig. 14. Histogram of the ratio $\text{NPDR}_{\text{EWA}}[k]/\text{NPDR}_{\text{DAA}}[k]$ in the presence of random clock drift; $\varepsilon_T = \varepsilon_\phi = 0.3$. Ratios greater than 1 are colored blue and ratios smaller than 1 are colored red.

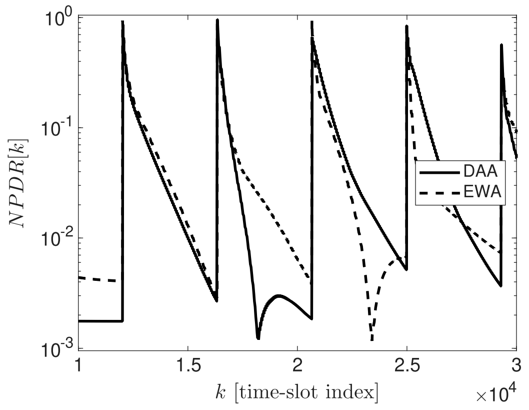


Fig. 15. $\mathrm{NPDR}_{\mathrm{DAA}}[k]$ and $\mathrm{NPDR}_{\mathrm{EWA}}[k]$ when clock phase and frequency resets are applied to 25% of the nodes in the network; $\varepsilon_T = \varepsilon_\phi = 0.3$.

D. Recovery From Random Phase and Frequency Resets

In typical operation of physical nodes, clocks' frequencies and phases may vary abruptly with temperature (especially in digitally compensated oscillators), [36], [37]. In this test we verify that synchronization can be quickly restored also when such frequency and phase jumps occur. To that aim we tested the DAA and the EWA for the network deployment in Fig. 2 in a scenario in which after convergence, a randomly selected set of 25% of the nodes experience random phase and frequency jumps. These jumps were implemented by generating the phases and frequencies of the selected nodes' oscillators via the statistics detailed in Sec. II, every 270 TDMA frames. Fig. 15 depicts the NPDRs for the DAA and the EWA. Observe that both schemes successfully re-acquire synchronization, whereas DAA is slightly superior to the EWA, as it achieves equal or smaller NPDR right before the reset is applied. In particular, this experiment demonstrates the robustness of the DAA to such resets. This robustness, in turn, implies that the learned weights are less affected by the initial phase and frequency differences between the nodes, implying that the propagation delays are the major factors affecting synchronization performance.

E. Robustness to Node Mobility

In the last test we examined how variations in the locations of the nodes affect synchronization accuracy. To that aim, we tested 10 randomly selected network deployments of strongly connected networks, enumerated as $NM1 - NM10$. For each network, after convergence, a group of 25% of the nodes was randomly selected, where each selected node was assigned a randomly selected direction. Then, each selected node started moving in its assigned direction at a speed of 150 [Km/h], traversing an overall distance of 1 [Km] during the simulation. Fig. 16 depicts the NPDRs for the DAA and the EWA in this experiment. It is clearly observed from the figure that changes in nodes' positions impact the NPDR for both the EWA and the DAA, which follows as both propagation delays and neighborhood sets vary due to node movement. Yet, both schemes exhibit gradual variation of NPDR values during the change in deployment. By computing the ratio of the NPDRs we observe that, without retraining, the DAA maintained

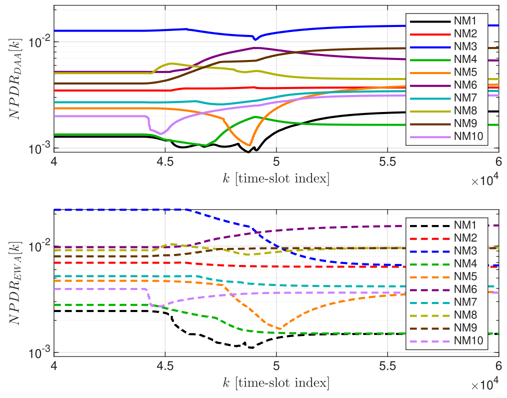


Fig. 16. $\mathrm{NPDR}_{\mathrm{DAA}}[k]$ (top, solid) and $\mathrm{NPDR}_{\mathrm{EWA}}[k]$ (bottom, dashed) when 25% of the nodes in the network begin moving at a speed of 150 [Km/h] in randomly selected directions, starting at $k = 44000$ and stopping at $k = 48000$; $\varepsilon_T = \varepsilon_\phi = 0.3$.

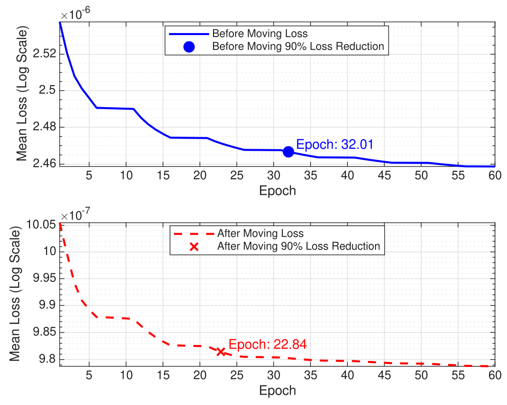


Fig. 17. Average of all loss functions for initial convergence (top, solid, blue) and for retraining after node movement (bottom, dashed, red) for scenario $NM1$. The markers show the point at which the loss completes 90% of its overall decrease; $\varepsilon_T = \varepsilon_\phi = 0.3$.

superiority over the EWA for displacements less than 800 [m], and for the overall displacement of 1 [Km] the DAA maintained its superiority in 70% of the tested scenarios. Evidently, Fig. 16 shows that the DAA exhibits significant robustness to node mobility.

One approach for mitigating the impact of large displacements on the DAA's performance is by retraining the network online. Fig. 17 presents the evolution of the average of the loss functions for all $N = 16$ node vs. the training epoch, for initial convergence (i.e., training before movement) and for the retraining after nodes have stopped, for scenario $NM1$. By comparing the time it takes for the loss function to achieve 90% of its convergent value (marked by the circle at the top plot and the 'X' at the bottom plot), we observe that the retraining is 30% faster than the initial convergence, which follows as only a partial update is required.

VII. Conclusion

In this work we have addressed distributed clock synchronization of both clock phases and periods in TDMA networks with HD transmissions. We proposed a synchronization scheme based on nested feedback loops that update the clock's phase and period using a weighted error. We analytically showed that the accuracy of this scheme depends on the unknown propagation delays, which motivated online learning for optimizing the weights according to the actual deployment. We then proposed a new algorithm which implements online learning via a DNN-based rule that weights the inputs with learned weights, determined by the measured time offsets and receive powers. The proposed training algorithm uses a novel initialization of the DNN parameters which facilitates initial convergence with DNN-based weights that approximate the EWA. Then, after a certain convergence time, each node acquires training data and subsequently applies training locally. We have shown that our new algorithm is superior to previously proposed schemes, including our previously proposed ESSBSA clock synchronization algorithm, and exhibits robustness to clock phase and frequency resets as well as to node mobility.

Appendix
Drift Due to Coupling When Phase and Frequency Are Simultaneously Updated

Let $k_N' = \lfloor k/N \rfloor \cdot N$ denote the most recent time index at which the correction signals $\Delta T_i[k]$ and $\Omega_i[k]$ in Eqns. (6) are updated w.r.t to time index k and let k_m denote the most recent time index at which node m transmitted at the previous cycle (i.e., the cycle whose last time instant was k_N'). Setting the weights $\alpha_{i,m}^{(\phi)} = \alpha_{i,m}^{(T)} = \alpha_{i,m}$, we recall that $\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} = 1$. Then, using the definition of $t_{i,j}[k]$ and the update rules (5) and (6) we can express

$$\begin{aligned}
&T_i[k+1] \\
&= T_i[k] + \frac{\epsilon_T}{N} \sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \Delta_T^{(i)}[m] \\
&\overset{(a)}{=} T_i[k] + \varepsilon_T \sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \big((\phi_m[k_m] - \phi_m[k_m - N]) \\
&\quad - (\phi_i[k_m] - \phi_i[k_m - N]) \big) / N^2 \\
&= T_i[k] + \frac{\varepsilon_T}{N^2} \sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \left(\left(\sum_{l=k_m-N}^{k_m-1} T_m[l] + \Omega_m\left[k_N' - 1\right] \right) \right. \\
&\quad \left. - \left(\sum_{l=k_m-N}^{k_m-1} T_i[l] + \Omega_i\left[k_N' - 1\right] \right) \right) \\
&\overset{(b)}{=} T_i[k] + \frac{\varepsilon_T}{N^2} \sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \left(\sum_{l=k_m-N}^{k_m-1} (T_m[l] - T_i[l]) \right) \\
&\quad + \frac{\varepsilon_T}{N^2} \left(\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \Omega_m\left[k_N' - 1\right] - \Omega_i\left[k_N' - 1\right] \right)
\end{aligned} \tag{25}$$

where (a) follows assuming that the propagation delay $q_{i,m}$ is fixed over an update period; and in (b) we used the fact that $\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} = 1$. Note that even if the period is synchronized across the times $k_m - N, k_m - N + 1, \dots, k_m - 1$, i.e., $T_m[l] = T_i[l]$, $l = k_m - N, k_m - N + 1, \dots, k_m - 1$ then still

$$T_i[k+1] = T_i[k] + \frac{\varepsilon_T}{N} \left(\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \Omega_m[k_N' - 1] - \Omega_i[k_N' - 1] \right).$$

Let k_l'' denote the time at which node l transmitted in the cycle of N transmissions whose last time instant is $k_N' - 1$ (i.e., the cycle preceding that to which k_m belongs). Then, we can write

$$\begin{aligned}
\Omega_m[k_N' - 1] &= \varepsilon_\phi \sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot \Delta_\phi^{(m)}[l] \\
&= \varepsilon_\phi \sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot \big(\phi_l[k_l''] + q_{m,l} - \phi_m[k_l''] \big).
\end{aligned}$$

Substituting this into the expression for $T_i[k+1]$ we obtain

$$\begin{aligned}
&T_i[k+1] \\
&= T_i[k] + \frac{\varepsilon_T}{N^2} \Bigg(\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \varepsilon_\phi \sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot (\phi_l[k_l''] + q_{m,l} - \phi_m[k_l'']) \\
&\qquad - \varepsilon_\phi \sum_{v\in\mathcal{N}(i)} \alpha_{i,v} \cdot \Delta_\phi^{(i)}[v] \Bigg) \\
&= T_i[k] + \frac{\varepsilon_T \varepsilon_\phi}{N^2} \Bigg(\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \Bigg(\sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot \big(\phi_l[k_l''] + q_{m,l} \\
&\qquad - \phi_m[k_l''] \big) - \Delta_\phi^{(i)}[m] \Bigg) \Bigg) \\
&= T_i[k] + \frac{\varepsilon_T \varepsilon_\phi}{N^2} \Bigg(\sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \Bigg(\sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot \big(\phi_l[k_l''] + q_{m,l} \\
&\qquad - \phi_m[k_l''] \big) - (\phi_m[k_m''] + q_{i,m} - \phi_i[k_m'']) \Bigg) \Bigg).
\end{aligned}$$

Note that even if all clock phases are synchronized at a given instant, the period would drift, i.e., substituting $\phi_l[k] = \phi_m[k]$ for all l, m, k in a certain cycle, we obtain

$$T_i[k+1] = T_i[k] + \frac{\varepsilon_T \varepsilon_\phi}{N^2} \sum_{m\in\mathcal{N}(i)} \alpha_{i,m} \left(\sum_{l\in\mathcal{N}(m)} \alpha_{m,l} \cdot q_{m,l} - q_{i,m} \right).$$

From the derivation above it is evident that even if all the clocks are synchronized, updating the phase and the period simultaneously will cause the periods to become unsynchronized.

References

[1] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks: A survey,” *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul. 2004.

[2] A. K. Karthik and R. S. Blum, “Recent advances in clock synchronization for packet-switched networks,” *Found. Trends Signal Process.*, vol. 13, no. 4, pp. 360–443, 2020.

[3] *IEEE Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, Standard 802.1AS-2011, 2011.

[4] R. Pigan and M. Metter, *Automating With PROFINET: Industrial Communication Based on Industrial Ethernet*. Hoboken, NJ, USA: Wiley, 2008.

[5] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, “Distributed synchronization in wireless networks,” *IEEE Signal Process. Mag.*, vol. 25, no. 5, pp. 81–97, Sep. 2008.

[6] E. Koskin, D. Galayko, O. Feely, and E. Blokhina, “Generation of a clocking signal in synchronized all-digital PLL networks,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 6, pp. 809–813, Jun. 2018.

[7] R. T. Rajan and A.-J. van der Veen, “Joint ranging and clock synchronization for a wireless network,” in *Proc. 4th IEEE Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process. (CAMSAP)*, Dec. 2011, pp. 297–300.

[8] X. Huan, W. Chen, T. Wang, H. Hu, and Y. Zheng, “A one-way time synchronization scheme for practical energy-efficient LoRa network based on reverse asymmetric framework,” *IEEE Trans. Commun.*, vol. 71, no. 11, pp. 6468–6481, Nov. 2023.

[9] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. 5th Symp. Operating Syst. Design Implement. (OSDI)*, vol. 36, Boston, MA, USA, Dec. 2002, pp. 147–163.

[10] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. sensor Syst.*, Nov. 2004, pp. 39–49.

[11] E. Abakasanga, N. Shlezinger, and R. Dabora, "Unsupervised deep-learning for distributed clock synchronization in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 12234–12247, Sep. 2023.

[12] O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP J. Wireless Commun. Netw.*, vol. 2007, no. 1, pp. 1–13, Jun. 2007.

[13] W. Jaafar, S. Naser, S. Muhaidat, P. C. Sofotasios, and H. Yanikomeroglu, "Multiple access in aerial networks: From orthogonal and non-orthogonal to rate-splitting," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 372–392, 2020.

[14] J. Zhu and S. S. Kia, "A SPIN-based dynamic TDMA communication for a UWB-based infrastructure-free cooperative navigation," *IEEE Sensors Lett.*, vol. 4, no. 7, pp. 1–4, Jul. 2020.

[15] J. C. López-Ardao, R. F. Rodríguez-Rubio, A. Suárez-González, M. Rodríguez-Pérez, and M. E. Sousa-Vieira, "Current trends on green wireless sensor networks," *Sensors*, vol. 21, no. 13, p. 4281, 2021.

[16] L. Xu, S. Sun, K. V. Mishra, and Y. D. Zhang, "Automotive FMCW radar with difference co-chirps," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 6, pp. 8145–8165, Dec. 2023.

[17] P. Chatterjee, J. A. Nanzer, and M. Yan, "Frequency consensus for distributed antenna arrays with half-duplex wireless coordination," in *Proc. IEEE Int. Symp. Antennas Propag. North Amer. Radio Sci. Meeting*, Jul. 2020, pp. 1585–1586.

[18] J. Du and Y.-C. Wu, "Distributed clock skew and offset estimation in wireless sensor networks: Asynchronous algorithm and convergence analysis," *IEEE Trans. Wireless Commun.*, vol. 12, no. 11, pp. 5908–5917, Nov. 2013.

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

[20] D. Märzinger, B. Etzlinger, P. Peterseil, and A. Springer, "Time-multiplexed AoA estimation and ranging," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2023, pp. 1–7.

[21] R. Fan, W. Liu, M. Li, and Z. Chai, "Clock offset and skew estimation based on correlation detection with one-way dissemination in wireless sensor networks," in *Proc. 10th Int. Workshop Signal Design Appl. Commun. (IWSDA)*, Aug. 2022, pp. 1–5.

[22] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[23] A. Giridhar and P. R. Kumar, "Distributed clock synchronization over wireless networks: Algorithms and analysis," in *Proc. 45th IEEE Conf. Decis. Control*, Dec. 2006, pp. 4915–4920.

[24] I. Zino, R. Dabora, and H. V. Poor, "Model-based learning for network clock synchronization in half-duplex TDMA networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2024, pp. 1618–1624.

[25] P. Palà-Schönwälder, J. Bonet-Dalmau, A. López-Riera, F. X. Moncunill-Geniz, F. del Águila-López, and R. Giralt-Mas, "Superregenerative reception of narrowband FSK modulations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 3, pp. 791–798, Mar. 2015.

[26] N. Naik, "LPWAN technologies for IoT systems: Choice between ultra narrow band and spread spectrum," in *Proc. IEEE Int. Syst. Eng. Symp. (ISSE)*, Oct. 2018, pp. 1–8.

[27] A. Korman and R. Vacus, "Distributed alignment processes with samples of group average," *Trans. Control Netw. Syst.*, vol. 10, no. 2, pp. 960–971, Jun. 2023.

[28] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.

[29] G. Garrigos and R. M. Gower, "Handbook of convergence theorems for (stochastic) gradient methods," 2023, *arXiv:2301.11235*.

[30] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Apr. 2019.

[31] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proc. Amer. Control Conf.*, vol. 2, Jun. 2003, pp. 951–956.

[32] H. Y. Kim, "Modeling and tracking time-varying clock drifts in wireless networks," Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, USA, Aug. 2014.

[33] D. Tulone, "A resource-efficient time estimation for wireless sensor networks," in *Proc. Joint Workshop Found. Mobile Comput.*, Oct. 2004, pp. 52–59.

[34] J.-S. Kim, J. Lee, E. Serpedin, and K. Qaraqe, "Robust clock synchronization in wireless sensor networks through noise density estimation," *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3035–3047, Jul. 2011.

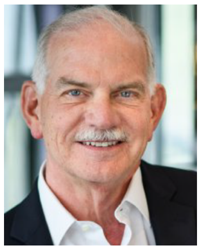
[35] D. Gerosa, R. Hou, V. Björk, U. Gustavsson, and T. Eriksson, "Autoregressive stochastic clock jitter compensation in analog-to-digital converters," 2025, *arXiv:2505.05030*.

[36] M. E. Frerking, *Crystal Oscillator Design and Temperature Compensation*. New York, NY, USA: Van Nostrand Reinhold Company, 1978.

[37] J. R. Vig, "Quartz crystal resonators and oscillators for frequency control and timing applications: A tutorial," Electron. Technol. Devices Lab, Fort Monmouth, NJ, USA, Tech. Rep. SLCET-TR-88-1 (Rev. 6.1), May 1993.

Itay Zino received the B.Sc. degree from the Sami Shamoon College of Engineering (SCE), Israel, in 2020, and the M.Sc. degree from the Ben-Gurion University of the Negev, Israel, in 2024, where he is currently pursuing the Ph.D. degree. His research interests include signal processing for communications, algorithm design, and electro-optical communications. In addition, he serves as a consultant in these fields for high-tech companies.

Ron Dabora (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Tel-Aviv University in 1994 and 2000, respectively, and the Ph.D. degree in electrical engineering from Cornell University, USA, in 2007. From 1994 to 2000, he was with the Ministry of Defense of Israel, and from 2000 to 2003, he was with the Algorithms Group, Millimetrix Broadband Networks, Israel. From 2007 to 2009, he was a Post-Doctoral Researcher with the Department of Electrical Engineering, Stanford University, USA. Since 2009, he has been with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel, where he is currently an Associate Professor. During the academic year 2022–2023, he was a Visiting Fellow at the Department of Electrical Engineering, Princeton University, USA. His research interests include network information theory, wireless communications, power line communications, and machine learning. He served as a TPC Member in international conferences, including WCNC, PIMRC, and ICC. From 2012 to 2014, he was an Associate Editor of IEEE SIGNAL PROCESSING LETTERS and from 2014 to 2019, he was a Senior Area Editor of IEEE SIGNAL PROCESSING LETTERS.



H. Vincent Poor (Life Fellow, IEEE) received the Ph.D. degree in EECS from Princeton University in 1977. From 1977 to 1990, he was on the faculty of the University of Illinois at Urbana–Champaign. Since 1990, he has been on the faculty at Princeton University, where he is currently the Michael Henry Strater University Professor. From 2006 to 2016, he was the Dean of Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Caltech. His research interests include information theory, machine learning, and network science, and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the book *Machine Learning and Wireless Communications* (Cambridge University Press, 2022). He is a member of the National Academy of Engineering and the National Academy of Sciences and is a foreign member of the Royal Society and other national and international academies. He received the IEEE Alexander Graham Bell Medal in 2017.