Unsupervised Deep-Learning for Distributed Clock Synchronization in Wireless Networks

Emeka Abakasanga, Nir Shlezinger, *Member, IEEE*, and Ron Dabora, *Senior Member, IEEE*

***Abstract*—One of the major factors which limits the throughput in wireless communications networks is the accuracy of time synchronization between the nodes in the network. Synchronization methods based on pulse-coupled oscillators (PCOs) have the advantage of simple implementation and achieve high accuracy when the nodes are closely located. However, such schemes tend to have poor synchronization performance for distant nodes, as well as in the presence of clock frequency offsets between the nodes. In this paper we present a novel PCO-based Deep neural network (DNN)-Aided Synchronization Algorithm coined DASA. We design DASA as a novel low-complexity and interpretable architecture by converting classic PCO-based synchronization into *a trainable discriminative model*. To enable DASA to operate in dynamic settings, we propose a novel, unsupervised, distributed, fast *online* training scheme which is able to train DASA *within a few sampling instances, locally*, thereby avoiding the need for information exchange between the nodes or for a central node for coordination. DASA is demonstrated to achieve an improvement by a factor greater than ten compared to the classic reference scheme. Lastly, we propose another novel, distributed *offline* training scheme for DASA, which is demonstrated to offer a tradeoff between performance and simplicity of deployment compared to the online training scheme, yet, at the same time, DASA with offline training still achieves superior performance compared to the classic reference scheme.**

***Index Terms*—Distributed clock synchronization, machine learning, model-based learning, pulse-coupled oscillators, unsupervised deep learning.**

I. Introduction

TIME synchronization is a critical prerequisite for the successful operation of wireless communications networks relying on time division multiple access (TDMA) to facilitate resource sharing. This follows as with TDMA, each node is allocated a time slot for transmission, thus, time synchronization across the nodes is essential to guarantee that there are no collisions, facilitating maximization of the spectral efficiency [1], [2].

Manuscript received 19 October 2022; revised 28 February 2023; accepted 17 April 2023. Date of publication 21 April 2023; date of current version 19 September 2023. This work was supported in part by the Israel Science Foundation under Grant 584/20 and in part by the Israeli Ministry of Economy via the 5G-WIN Consortium. The review of this article was coordinated by Prof. Xianbin Wang. *(Corresponding author: Ron Dabora.)*

Emeka Abakasanga and Nir Shlezinger are with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Be'er-Sheva 8410501, Israel (e-mail: abakasan@post.bgu.ac.il; nirshl@bgu.ac.il).

Ron Dabora is with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Be'er-Sheva 8410501, Israel, and also with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: daborona@bgu.ac.il).

Digital Object Identifier 10.1109/TVT.2023.3269381

There exist various approaches for achieving synchronization. These approaches can be classified as either local synchronization, in which the nodes mutually coordinate their clocks with their neighbours, or global synchronization, in which all nodes are synchronized to a global clock. In ad-hoc wireless networks, it is typically preferable for the nodes to synchronize in a distributed manner, such that the nodes in the network obtain and maintain the same network clock time based on local interactions and independent processing at the nodes, without requiring direct communications with a global synchronization reference [3], such as the Global Positioning System (GPS). Such architecture is also referred to as *distributed synchronization*. Traditional distributed time synchronization algorithms require periodic transmission and reception of time information, which is commonly implemented via packets containing a time-stamp data, exchanged between the coupled nodes [4], and is accordingly referred to as packet-based coupling. Packet-based synchronization has been extensively studied for wired and wireless networks, [5], with proposed protocols including the flooding time synchronization protocol [6], Precision Time Protocol [7], Network Time Protocol [8], generalized Precision Time Protocol [9], and Precision Transparent Clock Protocol [10, Sec. 3.5]. These protocols differ in the the way the time-stamp information is encoded, conveyed and processed across the nodes. The major drawbacks of packet-based coupling are the inherent unknown delays in packet formation, queuing at the MAC layer, and packet processing at the receiver. These delays could potentially make the received time-stamp carried by the packet, outdated, after processing is completed. Another significant drawback of packet-based synchronization is the high energy consumption due to the associated processing [3].

An alternative approach to packet-based synchronization, is physical-layer synchronization, which offers lower energy consumption and simpler processing, due to utilizing the broadcast nature of the wireless medium. In this approach, the time information is represented by the time at which a waveform transmitted by a sender node is received at each of the other nodes, hence, avoiding the inherently complex packet processing at the MAC layer and at the receiver [3]. A major approach for physical-layer synchronization is based on pulse-coupled oscillators (PCOs), which use the reception times of the pulses transmitted by the other nodes to compute a correction signal applied to adjust the current node's time [3], [11], [12]. In classic PCO-based synchronization [3], the correction signal is determined based on the output of a phase discriminator (PD) which computes the differences between the node's own time

and the reception times of the pulses from the other nodes. Each difference is weighted according to its corresponding relative received signal power w.r.t. the sum of the powers of the signals received at the node from the other nodes. While with this intuitive weighting, PCO-based synchronization is very attractive for wireless networks, the resulting synchronization performance significantly degrade in network configurations in which there are large propagation delays and clock frequency differences, and generally, full clock synchronization (frequency and phase) is not attained by current PCO-based schemes, see, e.g., [3]. *This motivates the design of a robust PCO-based time synchronization scheme, which is the focus of the current work.*

Main Contributions: In this work we propose a PCO-based time synchronization scheme which is robust to propagation delays and to clock frequency offsets between the nodes. The inherent challenge in this problem is the mapping of the output of the PD into a voltage controlled clock (VCC) correction signal. We address this challenge by incorporating a deep neural network (DNN) into this mapping, as such networks are known to be able to learn complex functions from data. We use a model-based deep learning approach (see, e.g., [13], [14]) where the DNN generates the weights used by the PCO-based architecture, achieving both improved robustness and reduced complexity, along with improved synchronization accuracy. Accordingly, *the first contribution of this work is the design of a model-based deep neural network-aided synchronization algorithm (DASA) for PCO-based network clock synchronization.* The proposed architecture converts the classic synchronization method described in [3] (previously discussed in [15], see Section II-B) into a *trainable discriminative model* [16] while generating its weights using a dedicated attention model that follows the operation of [3]. DASA is demonstrated to be very robust to clock resets and to node mobility.

The application of DNNs to communications and signal processing tasks is quite common, and has been considered in multiple research works. Yet, the dynamic nature of the wireless channels, their incomplete statistical characterization and the limited resources available at wireless nodes make the problem of learning the DNN's parameters quickly and efficiently at a low computational complexity, a major challenge that stands in the way of widely implementing DNNs for improving performance of communications and signal processing tasks in wireless networks scenarios.

Hence, *the second major contribution of our work is the design of a novel unsupervised, distributed, fast training procedure for the proposed DASA*. In particular, we construct a new *decentralized loss measure*, that facilitates rapid convergence. The unsupervised nature of the training procedure avoids the need to a-priori generate training data and facilitates *online training* at the nodes. Thanks to its online nature, training accounts for the *actual geographic deployment and physical parameters* of the nodes, which leads to *a significant performance improvement*. Furthermore, as the loss measure is designed to be computed at each node locally, then *the need for exchanging information between the nodes during the training process is avoided*, yet an overall coordinated synchronization across the nodes is achieved. It is also clarified that the proposed local training method *differs* from deep reinforcement learning (DRL) approaches as there is no repeated interaction with the environment, which follows as during the data acquisition phase, the weights of the DNNs are not updated.

Simulation results show that thanks to our novel training process, the proposed scheme achieves an improvement of at least an order of magnitude in synchronization accuracy across the nodes, compared to the classic reference scheme. Another important aspect we incorporate in our scheme is handling of unknown, possibly time varying network topology, which arises because the input and output spaces of each DNN have to be set and adapted in real time to be equal to the number of nodes received above the receiver's detection threshold. Note that temporally varying input and output spaces are an uncommon consideration in works implementing DNNs. This consideration is addressed in Section IV-A.

In certain deployments, it may still be desirable to operate the synchronization algorithm immediately, without online training, e.g., to save power or in case of very short communications, such as for emergency applications and machine-to-machine communications. Therefore, the *third contribution of this work is the design of an offline training scheme based on simulated data*. The proposed offline training method, which is complementary to the online scheme, can be applied to train the DNNs at the nodes based only statistical deployment and propagation information. It is demonstrated that offline training offers a tradeoff between performance and simplicity of deployment, yet its performance is still better than that of the classic reference scheme.

Organization: The rest of this work is organised as follows: Section II reviews the fundamental structure of PCO-based synchronization schemes. Section III presents the problem formulation, highlights the weaknesses of the classic weighting rule and states the objective of this work. Subsequently, Section IV presents our proposed DASA, together with the novel online and the offline training procedures, highlighting their novel aspects. Numerical examples and discussions are provided in Section V. Lastly, Section VI concludes this work.

Notations: In this paper, deterministic column vectors are denoted with boldface lowercase letters, e.g., $\mathbf{x}$, deterministic scalars are denoted via standard lowercase fonts, e.g., x, and sets are denoted with calligraphic letters, e.g., $\mathcal{X}$. We denote the sets of real numbers, positive integers and of integers by $\mathcal{R}$, $\mathcal{N}$ and $\mathcal{Z}$, respectively. Lastly, all logarithms are taken to base-2.

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

A. Network and Clock Models

We study discrete-time (DT) clock synchronization for wireless networks, considering a network with N nodes, indexed by $i \in \{1, 2, \ldots, N\} \triangleq \mathcal{I}_N$. Each node has a clock oscillator with its own inherent period, denoted by $T_i \in \mathcal{R}^{++}$, $i \in \mathcal{I}_N$. Ignoring phase noise, see, e.g., [17, Sec. V], [3], then, at update index $k \in \mathcal{Z}$ the clock time $\phi_i(k)$, at node $i \in \mathcal{I}_N$,

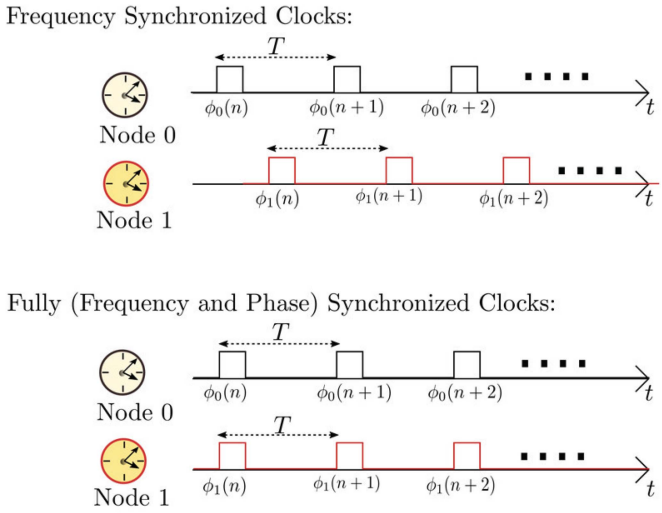


Fig. 1. Illustration of clock frequency synchronization and of full clock synchronization, for a pair of nodes.

can be expressed as

$$\phi_i(k) = \phi_i(0) + k \cdot T_i, \tag{1}$$

where $\phi_i(0)$ is the initial clock time at index $k = 0$.

In a wireless network, when the clock periods of the different nodes, T_i, $i \in \mathcal{I}_N$, are different, then the nodes' transmissions may overlap in time and frequency (a situation which is referred to as "collision"), resulting in loss of information. Moreover, even when the clock periods are all identical, referred to as *clock frequency synchronization*, a time offset (also referred to as *phase offset*) between the clocks may exist, as illustrated in Fig. 1 (see [3, Fig. 2] for additional elaboration), which again results in collisions. To facilitate high speed communications, the nodes must synchronize both their *clock frequencies as well as their clock phases* to a common time base. This is referred to as *full clock synchronization*. To that aim, the nodes in the network exchange their current time stamps, and based on the exchanged time information, they attempt to reach a consensus on a common clock.

Lastly, we note that it is commonly assumed that node locations are static and the propagation channels are time-invariant, see, e.g., [3]. The case of time-varying communications links has been studied in [18], [19], [20], assuming the links between the nodes randomly evolve over time, and each node subsequently updates its parameters according to the information received from the other nodes. In [18], necessary conditions for convergence were established by combining graph theory and system theory for bidirectional and for unidirectional links. It was concluded in [18] that synchronization could fail, even for fully-connected networks.

B. Pulse-Coupled Synchronization

In pulse coupled synchronization, each node $i \in \mathcal{I}_N$ broadcasts a sequence of synchronization signatures, which uniquely identify the transmitting node, e.g., [21], [22], [23], where transmission times are determined at each transmitting node according to its own local clock. Each receiving node processes all the synchronization signatures it receives above its detection threshold, namely, it processes all the synchronization signatures carried by signals whose received powers were sufficiently high at the receiving node, such that it was able to reliably determine the presence of a signature, associate it with a transmitting node, and determine its corresponding reception time. Then, based on the processed signatures, the receiver updates its local clock.

Fig. 2 depicts the distributed pulse-coupled phase-locked loop (PLL). It is commonly assumed that the nodes can transmit and receive at the same time. Let $t_{i,j}(k)$ denote the time at node i at which the k'th stamp from node j is received, $\alpha_{i,j}$ denote the weight at node i for the timing information received from node j, and let $\Delta\phi_{i,j}(k) \triangleq t_{i,j}(k) - \phi_i(k)$. As depicted in Fig. 2, the PLL is a feedback loop, which consists of a phase difference detector (PDD), a linear, time-invariant (LTI) filter with a transfer function $\varepsilon(z)$, and a VCC. Mathematically, the output of the PDD at the k'th clock cycle, $k \in \mathcal{N}$, at the i'th node, denoted by $\Delta\phi_i(k)$, is expressed as (see [3]):

$$\begin{aligned}\Delta\phi_i(k) &= \sum_{j=1, j\neq i}^{N} \alpha_{i,j} \cdot (t_{i,j}(k) - \phi_i(k)) \\ &= \sum_{j=1, j\neq i}^{N} \alpha_{i,j} \cdot \Delta\phi_{i,j}(k),\end{aligned} \tag{2}$$

where $\alpha_{i,j} \in [0, 1]$. Letting $q_{i,j}(k)$ denote the propagation time between node i and node j, then at update number k, $t_{i,j}(k) = \phi_j(k) + q_{i,j}(k)$. The PDD output is fed into a loop filter $\varepsilon(z)$, whose output drives the VCC that re-calibrates the instantaneous time at the i'th node.

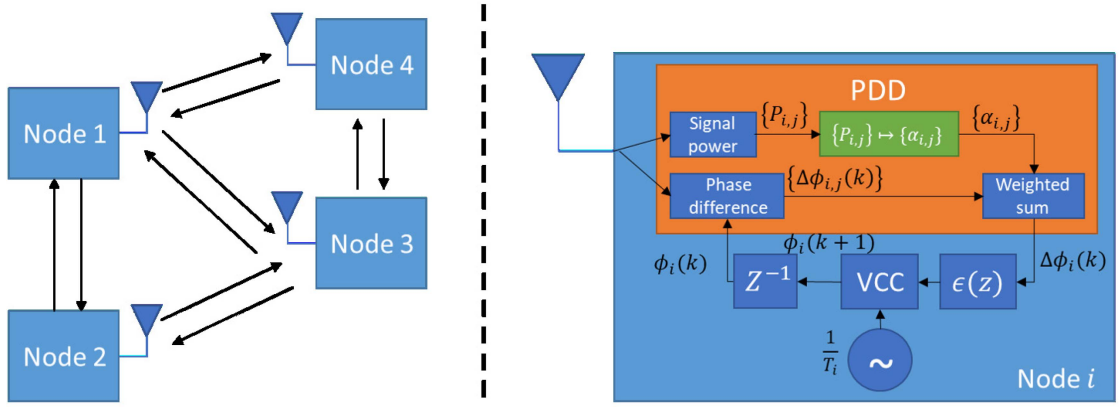
In the classic PCO-based PLL design of [3], $\alpha_{i,j}$ corresponds to the relative signal power of the pulse received at node i from node j: Letting $P_{i,j}$ denote the power of the pulse received at node i from node j, then, in the classic algorithm of [3], $\alpha_{i,j}$ is computed as [3], [24], [25]:

$$\alpha_{i,j} = \frac{P_{i,j}}{\sum_{j=1, j\neq i}^{N} P_{i,j}}. \tag{3}$$

From (3) it follows that the value of $\alpha_{i,j}$ depends on the factors which affect the received power levels. These include the distance between the nodes, which is the most dominant factor, as well as shadowing and fading. When implementing a first-order PLL, then $\varepsilon(z)$ is set to $\varepsilon(z) = \varepsilon_0$, and the update rule can be stated as (see [3, Eqns. (16), (23)]):

$$\phi_i(k+1) = \phi_i(k) + T_i + \varepsilon_0 \cdot \sum_{j=1, j\neq i}^{N} \alpha_{i,j} \cdot \Delta\phi_{i,j}(k). \tag{4}$$

We refer to the rule (4) with weights (3) as the *classic algorithm* or the *analytic algorithm*. This rule (with (3)) has been acceptable as a baseline in multiple works on network clock synchronization, e.g., [15], [26], [27], hence, we use it as a baseline also in the current work. It has been noted in [3] that for pulse-coupled first-order DT PLLs, synchronization can be achieved if and only if for any node pair there exists a connecting



(a) (b)

Fig. 2. (a) Sample topological deployment of $N = 4$ pulse-coupled nodes; (b) Structure of the synchronization algorithm based on the classic model (4), see [3].

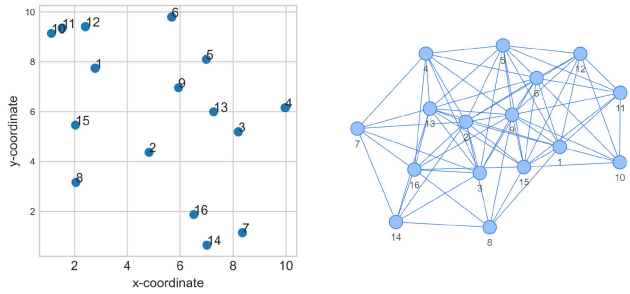
path. The connection between each node pair need not be direct, and may also run via intermediate nodes, [28].

The rule in (4) was expressed as a time-invariant difference equation in [3], [15], for which the steady-state phase expressions at the nodes, in the limit as k increases to infinity, were derived. It was shown that for the case of *no propagation delay* and *identical clock periods* at all nodes, i.e., $q_{i,j}(k) = 0$, $i, j \in \mathcal{I}_N$, $k \in \mathcal{Z}$, and $T_i = T_{\text{nom}}$, $i \in \mathcal{I}_N$, the rule (4) generally results in the network attaining full synchronization. On the other hand, when there are propagation delays and/or different clock periods at the nodes, then typically, frequency synchronization to a common frequency is attained, but full synchronization is not attained, and in order to decrease phase synchronization mismatch, the order of the loop has to be increased. Yet, at the same time, such an approach reduces the stability margin of the loop, and increases its complexity.

In this paper we focus on the common and more practical scenario in which there are propagation delays and different clock periods, which generally results in asynchronous clocks at steady state. Accordingly, the objective of our algorithm is to attain full synchronization for this important general scenario.

III. Problem Formulation

Consider a network with N nodes, such that each node $i \in \mathcal{I}_N$ has its own clock with a period of T_i, where generally $T_i \neq T_j$ for $i \neq j$, and let $\phi_i(k)$ denote the clock time at update number $k \in \mathcal{N}$. At startup, the clock at each node $\phi_i(0)$, $i \in \mathcal{I}_N$, is generated as a random value in the range $[0, T_i)$ [3]. The nodes are located at geographically separate locations, where node i is located at coordinate (x_i, y_i), and we let $d_{i,j}$ denote the distance between nodes i and j. Assuming line-of-sight propagation, a signal transmitted from node i is received at node j after $\frac{d_{i,j}}{c}$ time units, where c is the speed of light. We assume that the nodes are not aware of their relative locations and of the clock periods at the other nodes. *The objective is to synchronize the clock times* $\{\phi_i(k)\}_{i \in \mathcal{I}_N}$ *such that at steady-state, at each* k*, it holds that* $\phi_i(k) = \phi_j(k)$, $\forall i \neq j$.



(a) (b)

Fig. 3. (a) Geographic locations of the nodes in the wireless network considered in Section III; (b) Network graph showing the connected node pairs.

As a motivating scenario, consider a wireless network with $N = 16$ nodes located in a square area of dimensions 10 [Km] $\times$ 10 [Km], with locations depicted in Fig. 3. In this example, the startup value for the clock of node i, $\phi_i(0)$, is generated uniformly over $[0, T_i)$, see, e.g., [21], [26], [29], [30]. Each node transmits periodically at its corresponding clock times, and processes the pulse timings for its received pulses using the DT PLL update rule of [3, Eqns. (16), (23)] to synchronize their clocks, see (4). For the purpose of the numerical evaluation, we let the nominal period of the clocks in the network, denoted T_{nom}, be $T_{\text{nom}} = \frac{1}{200}$ [sec]. The period T_i for the clock at node i is obtained as by randomly generating clock periods with a maximum deviation of 100 [ppm]:

$$T_i = T_{\text{nom}} \cdot (1 + B_0 \cdot 10^{-A}), \tag{5}$$

where B_0 is a uniformly distributed random variable whose value is either 1 or -1, and A is uniformly selected from the interval

[4,6]. For time-invariant channels, the corresponding propagation delays are given by $q_{i,j}(k) = q_{j,i}(k) = \frac{d_{i,j}}{c}, \forall i, j \in \mathcal{I}_N$ and $k \in \mathcal{N}$.

For simplicity, all nodes have the same transmit power of $P_t = 33$ [dBm] (different powers can be modeled using different topologies), and we use the two-ray wireless propagation model with isotropic antennas at the nodes, i.e., the antenna gain is $G_i = 1$ at all directions, $\forall i \in \mathcal{I}_N$. For node heights of 1.5 [m], it follows that the received power at node i from node j, denoted $P_{i,j}$, is given by [31, Eqn. 2.1-8]:

$$P_{i,j} \approx \frac{P_t \cdot G_i G_j h_i^2 h_j^2}{d_{i,j}^4} = \frac{10}{d_{i,j}^4} = P_{j,i}. \tag{6}$$

We assume detection threshold (also referred to as receiver sensitivity) of -114 [dBm], [32], [33], [34] and as a result, 48 node pairs do not have direct reception. This is depicted in the graph in Fig. 3(b), in which nodes not connected by edges do not have a direct link. The evaluated received power levels are then applied for calculating the weights $\alpha_{i,j}$ via (3).

To showcase the shortcomings of existing approaches in such a realistic and challenging setting, we evaluated the performance of the rule [3, Eqns. (16), (23)] for a first-order PLL with $\varepsilon(z) = 1$ and for a second-order PLL with $\varepsilon(z) = \frac{1}{1-0.3z^{-1}}$. The period was observed to converge at time $k = 2799$ to a mean value of $T_{c,ANA}(2799) = 0.00500208$, and $T_{c,ANA_1}(2799) = 0.00500274$, respectively. The evolution, as k increases, of the modulus of the *clock times* w.r.t the synchronized periods $T_{c,ANA}(2799)$ and $T_{c,ANA_1}(2799)$ when $\varepsilon(z)$ corresponds to a first-order PLL and to a second-order PLL, respectively, is depicted in Fig. 4(a) and (b). Observe that indeed, for both PLLs, different phases are achieved at steady state. It follows that the algorithm [3, Eqns. (16), (23)] achieves frequency synchronization but *not full synchronization*, resulting in collisions between the transmissions of the different nodes, which, in turn, reduce the throughput of the communications network.

The examples above clearly demonstrate the motivation for this research: As with the ad-hoc analytic weights expression (3), synchronization performance, when propagation delays and/or clock period differences exist, are unsatisfactory, we propose a DNN-aided mechanism which learns to generate the weights at the nodes, such that full network clock synchronization is achieved with a zero-order loop, given a network scenario with a geographic deployment and clock mismatches, which are all *unknown at the nodes*. To facilitate practical implementation, we require that the algorithm's operation will be *distributed*, such that each node adapts its clock *independently*, processing only *its own received signals*. Note that without independent processing, the network throughput is further decreased due to the exchange of messages for facilitating coordinated processing. Our approach maintains the structure of the update rule (4), and focuses on replacing the weights $\alpha_{i,j}$ with weights learned using a DNN, where a *novel* training algorithm is proposed for training the DNN coefficients *based on local calculations at each node*. This approach is referred to as model-based learning. The rationale is that (3) does not account for clock frequency differences between the nodes and does not fully account for propagation

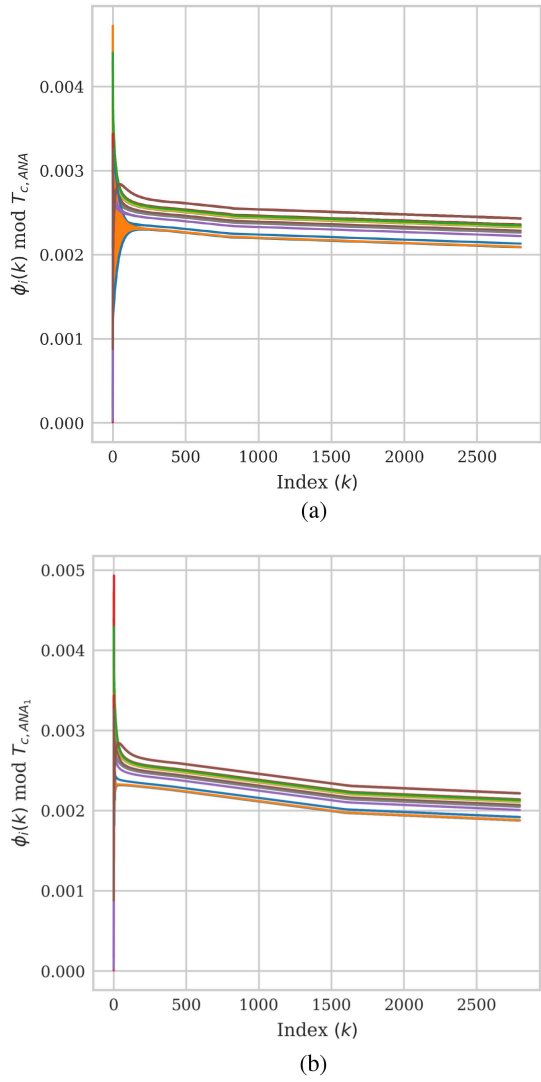


Fig. 4. Clock times $\phi_i(k)$ modulo the converged period at $k = 2799$, with $\alpha_{i,j}$'s given by (3), for the analytical rule of (4) with (a) a zero-order loop filter $\varepsilon(z) = 1$ and (b) a first-order loop filter $\varepsilon(z) = \frac{1}{1-0.3z^{-1}}$.

delays, thus, it is reasonable to expect a DNN trained with the actual propagation delays and clock frequency differences to better account for these factors. Model-based learning is also shown to carry the important advantage of robustness to the training set, as the learned functionality is restricted and does not involve end-to-end operation [13], [14], [35]. In the next section we detail our proposed DNN-based clock synchronization algorithm.

IV. A Novel DNN-Aided Distributed Clock Synchronization Algorithm

In this section we first describe the overall system in Section IV-A. Then, in Section IV-B we detail how the DASA can be trained *locally and independently* at each node in *an unsupervised manner*. Subsequently, in Section IV-C we propose an offline training scheme for the DASA. Lastly, we present

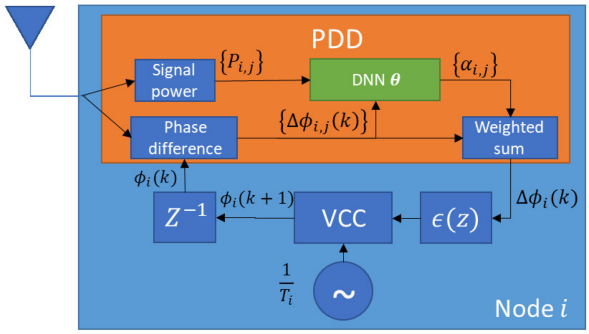


Fig. 5. Block diagram of the proposed DASA.

a discussion and a complexity analysis of the algorithm in Section IV-D.

A. PLL-Based Synchronization With Deep Learning

Our design is inspired by attention mechanisms, which are deep models that learn to compute input-dependent weighting coefficients as a means of assigning increased significance to certain input sets and a lesser significance to others. This approach was selected as it bears a clear similarity to the effect of the weighting coefficients $\alpha_{i,j}$ applied to the PDD output in (4). Attention models have shown significant empirical success in natural language and in time sequence processing [36]. Consequently, we replace the weights $\alpha_{i,j}$ of Eqn. (3) with a DNN-aided computation, where the inputs to the DNN consist of the received power levels and the relative time differences $\{\Delta\phi_{i,j}(k)\}$. The resulting DASA is schematically depicted in Fig. 5. Note that DASA preserves the structure of the model-based DT PLL synchronization scheme of (4), while incorporating a DNN to learn the mappings of the inputs into weighting coefficients, accounting for both the actual delays and the clock frequency differences between the nodes, which occur in practice.

We denote the DNN parameters at node i by $\boldsymbol{\theta}_i$ and use $\psi_{\boldsymbol{\theta}_i}(\cdot)$ to denote the corresponding mapping. For a given value of $\boldsymbol{\theta}_i$, the DNN at node i maps the $2(N-1)$ values $\{\Delta\phi_{i,j}(k), P_{i,j}(k)\}_{j=1,j\neq i}^{N}$ into the $N-1$ weighting coefficients which may vary with k. The weighted sum of $\Delta\phi_{i,j}(k)$ is then input to a loop filter having a transfer function $\varepsilon(z)$, and the output of the loop filter drives the VCC. The overall resulting time update rule can be expressed as

$$\phi_i(k+1) = \phi_i(k) + T_i + \varepsilon_0 \cdot \sum_{j=1,j\neq i}^{N} \left[\psi_{\boldsymbol{\theta}_i} \left(\{(\Delta\phi_{i,j'}(k), P_{i,j'}(k))\}_{\substack{j'=1,\\ j'\neq i}}^{N} \right) \right]_j \cdot \Delta\phi_{i,j}(k), \tag{7}$$

where $[\psi_{\boldsymbol{\theta}_i}(\cdot)]_j$ denotes the output of the DNN used for weighting the time difference between node i and node j at node i.

Handling Unknown and Varying DNN Input and Output Dimensions due to Receivers' Detection Thresholds: The fact that each receiver has a detection threshold below which it is not able to reliably detect the existence of a signal, has to be accounted for in the design of the DNN. As the geographical locations of the nodes are unknown at the other nodes, *the effect of the detection threshold has to be handled without a-priori knowledge at the nodes as to which are the nodes whose signals cannot be detected at each receiver*. Accordingly, it is not possible to a-priori set the number of inputs and outputs at each DNN to match the number of nodes received above the detection threshold. Furthermore, fixing the number of DNNs' inputs and outputs will not allow the nodes to handle mobility, as mobility varies nodes' connectivity and accordingly the input and output spaces of the DNNs at the nodes should vary. We thus set the *number of DNN inputs and outputs, each to* $2(N-1)$ at all nodes. Then, for transmitted signals that reach a receiver below its detection threshold, we set the corresponding input values of *receive power and phase difference* at the receiver to 0. This can be implemented, e.g., by noting that signatures of certain users were not detected during a clock update cycle. As the DNN outputs $N-1$ weights, then also the output weights corresponding to the timing of signals not observed within an update cycle are set to zero. For example, if the pulse transmitted at the k'th update cycle from node j is not detected at node i, then we set DNN inputs $P_{i,j}(k) = 0$ and $t_{i,j}(k) - \phi_i(k) = 0$, as well as set DNN output $[\psi_{\boldsymbol{\theta}_i}(\{(\Delta\phi_{i,j'}(k), P_{i,j'})\}_{j'=1,j'\neq i}^{N})]_j = 0$ in the calculation of the update.

We note that our proposed DNN is implemented as a multi-layered perceptron (MLP), instead of using more sophisticated trainable attention mechanisms (e.g., multi-head attention [36]). This follows from the fact that the network size N is assumed to be fixed. This facilitates utilizing DNNs which can learn to exploit arbitrary dependencies between the inputs, while having a relatively low computational complexity and being simple to train. Finally, the output of the MLP is guaranteed to constitute weighted averaging coefficients by applying a softmax output layer with $N-1$ outputs.

B. Online Training Scheme

In order for DASA to be robust and capable of coping with different environments and topologies, it has to rely on *unsupervised local training*. This facilitates training at each device locally, without requiring access to some ground-truth clock values. Here, we present an online training method for DASA, which allows it to adapt on device during synchronization, while Section IV-C presents a training procedure which is suitable for offline learning based on data corresponding to multiple settings. We next elaborate on the training procedure, and subsequently discuss how one can acquire the data for local training online.

1) Unsupervised Local Training: Since training is done in an unsupervised manner, the training set at each user is a sequence of N_T input sets, each consists of $N-1$ pairs of the receive time and received power level for the signals received at a node from all of the other nodes. Accordingly, the training data set for the i'th node is given by

$$\mathcal{D}_i = \left\{ \{(t_{i,j}(k), P_{i,j}(k))\}_{j=1,j\neq i}^{N} \right\}_{k=1}^{N_T}. \tag{8}$$

In the next subsection we detail how these measurements are obtained.

Algorithm 1: Unsupervised Online Local Training at Node i.

Data: Data set $\mathcal{D}_i$, learning rate μ, initial weights $\boldsymbol{\theta}_i$, period T_i, number of epochs E

for epoch $= 1$ *to* E **do**
  for $k = 1$ *to* N_T **do**
    Forward pass $\left\{\left(t_{i,j}(k), P_{i,j}(k)\right)\right\}_{j=1,j\neq i}^{N} \in \mathcal{D}_i$ to obtain $\phi_i(k+1)$ using Eqn. (7).
  end
  Compute loss $\mathcal{L}_{\mathcal{D}_i}(\boldsymbol{\theta}_i)$ via Eqn. (9);
  Compute gradient $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathcal{D}_i}(\boldsymbol{\theta}_i)$ using backpropagation through time;
  Update weights via $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \mu \cdot \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathcal{D}_i}(\boldsymbol{\theta}_i)$.
end

The data set in (8) *does not contain any ground-truth clock value*. Nonetheless, it can still be used for training the algorithm to minimize the *relative time differences*, i.e., the differences between each $t_{i,j}(k+1)$ and the clock time $\phi_i(k+1)$ produced by the DNN-aided system after processing $\{(t_{i,j}(k), P_{i,j}(k))\}_{j=1,j\neq i}^{N}$. Since we are interested in achieving fast convergence, then offsets at earlier time instances are more tolerable compared with those obtained at later values of k. Accordingly, we weight the relative time differences in the computation of the loss function by a monotonically increasing logarithmic function of k, see [37]. The resulting loss function is given by

$$\mathcal{L}_{\mathcal{D}_i}(\boldsymbol{\theta}_i) = \sum_{k=1}^{N_T} \log(k+1) \left(\sum_{\substack{j=1,\\ j\neq i}}^{N} \left(t_{i,j}(k+1) - \phi_i(k+1)\right)^2 \right), \tag{9}$$

with $\phi_i(k+1)$ computed recursively from $\phi_i(k)$ based on $\mathcal{D}_i$ and $\boldsymbol{\theta}_i$ via (7) with $\Delta\phi_{i,j}(k) = t_{i,j}(k) - \phi_i(k)$. The fact that the loss (9) is a quadratic function of $\phi_i(k+1)$, which in turn is a linear recursive function of the DNN output via (7), indicates that one can compute the gradient of the loss with respect to the weights via backpropagation through time [38]. Moreover, the loss (9) can be computed in an unsupervised manner by each node locally, hence, it facilitates unsupervised local training via conventional first-order based optimizers, e.g., mini-batch stochastic gradient descent (MB-SGD). The overall training algorithm when utilizing the conventional gradient descent optimizer is summarized as Algorithm 1.

2) Data Acquisition and DNN Training: The conventional practice in deep learning is to train DNNs offline, using pre-acquired data for training. This is inapplicable to clock synchronization in wireless networks as training should account for the specific propagation delays of the deployed network and clock frequency differences, then, to obtain the best synchronization performance, training should take place after deployment.

The training procedure in Algorithm 1 is particularly tailored to support on-device training, as it does not require ground-truth clock values and can be carried out locally. However, it still relies on providing each node with the training data set $\mathcal{D}_i$ in (8). The sets $\mathcal{D}_i$, $i \in \mathcal{I}_N$, can be acquired by simply having each node transmit a sequence of N_T synchronization signals. Then, each node forms its data set by collecting its receives signals' time stamps and powers. In particular, once the network is deployed and powered up, each device transmits N_T pulses, and uses its received measurements to form its local data set $\mathcal{D}_i$. This step is carried out when the nodes are not synchronized. During this data acquisition step, the nodes *do not update* their DNN coefficients, thus the parameters $\boldsymbol{\theta}_i$ at node i during this step are fixed to those set at the initialization. Then, at the local unsupervised training step, each node $i \in \mathcal{I}_N$ trains its local DNN via Algorithm 1, using the acquired data $\mathcal{D}_i$. The trained model coefficients $\boldsymbol{\theta}_i$ are then applied to compute the weights, replacing the $\alpha_{i,j}$'s of (3), *without requiring additional samples to be acquired and without re-training,* i.e., operating in a one-shot manner without inducing notable overheard. This local training method thus *differs* from DRL approaches, where training is carried out by repeated interaction.

C. Offline Training Scheme

While online training is designed to achieve the best performance as it accounts for the actual deployment characteristic, sometimes it is desirable to have the node synchronize immediately, without implementing a training at startup. To facilitate such operation we next consider training the DNNs offline over multiple topologies, instead of training online at startup for the actual deployment topology. To implement offline training, we use a set of N_{top} network topologies, each generated over the same network area of the planned deployment. For each topology instance, the individual nodes acquire *simulated training data*, generated using the propagation model appropriate for the network's deployment. The training data consists of N_T time intervals, where at each interval, each node collects $N-1$ inputs, each consisting of a pair of receive time and received power level as specified in (8). The training data for the i'th node over all the instances of network topologies is denoted as $\mathcal{D}_i^{N_{top}} = \{\mathcal{D}_{i,n_{top}}\}_{n_{top}=1}^{N_{top}}$, where $\mathcal{D}_{i,n_{top}}$ denotes the training data at node i for the n_{top}'th topology.

Loss Function: In online training, the loss function is computed for a single topology over a single batch, and only one step of gradient descent (GD) is applied at each epoch. In contrast, for the offline training we apply the MB-SGD. In this approach, the set $\mathcal{D}_i^{N_{top}}$ is divided into mini-batches of a fixed size denoted by N_{mb}, and the number of mini-batches is denoted by $N_{batch} = \frac{N_{top}}{N_{mb}}$. At each epoch, the MB-SGD at node i operates as follows:

1) A mini-batch is selected for training the node's DNN in a sequential order.
2) Estimate the average loss over the mini-batch topologies, where the loss for the n_{top}'th topology, denoted by $\mathcal{L}_{\mathcal{D}_{i,n_{top}}}(\boldsymbol{\theta}_i)$, is obtained via (9).
3) Compute the gradient and update the DNN's weights using the computed gradient.
4) Repeat steps 1–3 for all mini-batches.

The mini-batch training procedure is summarized in Algorithm 2, where $t_{i,j}^{n_{top}}(k)$, $P_{i,j}^{n_{top}}(k)$, and $\Delta\phi_{i,j}^{n_{top}}(k)$ denote the corresponding values for the n_{top}'th topology.

Algorithm 2: Unsupervised Offline Local Training at Node i.

Data: Data set $\mathcal{D}_i^{N_{top}}$, learning rate μ, initial weights $\boldsymbol{\theta}_i$, period T_i, number of epochs E

for epoch $= 1$ *to* E **do**
  Randomly shuffle $\mathcal{D}_i^{N_{top}}$;
  for btch $= 0$ *to* $N_{batch} - 1$ **do**
    for $k = 1$ *to* N_T **do**
      Forward pass
      $\left\{\left\{\left(t_{i,j}^{n_{top}}(k), P_{i,j}^{n_{top}}(k)\right)\right\}_{j=1, j\neq i}^{N}\right\}_{n_{top}=\text{btch}\cdot N_{mb}+1}^{(\text{btch}+1)\cdot N_{mb}} \in \mathcal{D}_i^{N_{top}}$ to obtain $\left\{\psi_{\boldsymbol{\theta}_{i,n_{top}}}(\Delta\phi_{i,j}^{n_{top}}(k), P_{i,j}^{n_{top}}(k))\right\}_{n_{top}=\text{btch}\cdot N_{mb}+1}^{(\text{btch}+1)\cdot N_{mb}}$;
      Evaluate $\left\{\phi_{i,n_{top}}(k+1)\right\}_{n_{top}=\text{btch}\cdot N_{mb}+1}^{(\text{btch}+1)\cdot N_{mb}}$ applying Eqn. (7) for each n_{top};
    end
    Compute loss $\mathcal{L}_{\mathcal{D}_i}(\text{btch}, \boldsymbol{\theta}_i) \triangleq \frac{1}{N_{mb}} \sum_{n_{top}=\text{btch}\cdot N_{mb}+1}^{(\text{btch}+1)\cdot N_{mb}} \mathcal{L}_{\mathcal{D}_{i,n_{top}}}(\boldsymbol{\theta}_i)$, via Eqn. (9);
    Compute gradient $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathcal{D}_i}(\text{btch}, \boldsymbol{\theta}_i)$ using backpropagation through time;
    Update weights via $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \mu \cdot \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathcal{D}_i}(\text{btch}, \boldsymbol{\theta}_i)$.
  end
end

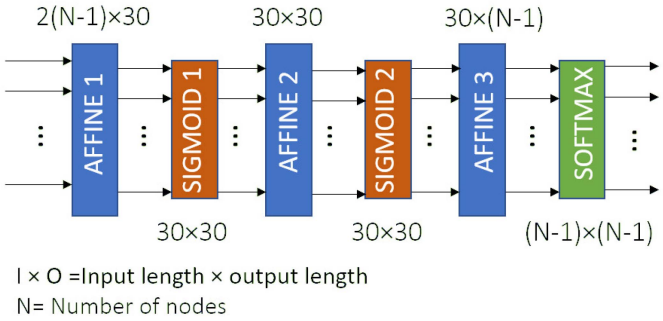


Fig. 6. Block diagram of the DNN utilized by the proposed DASA in the experimental study reported in Section V.

D. Discussion and Complexity Analysis

The proposed DASA uses unsupervised distributed training to map the measured inputs into weighting coefficients, which are applied in PCO-based synchronization, in order to overcome the inherent sensitivity to non-negligible propagation delays and to clock frequency differences of the classic algorithm [3]. DASA supports the usage of compact, low-complexity DNNs, that are applicable to hardware-limited wireless devices. For instance, in our numerical study reported in Section V, we utilized the simple three-layer MLP illustrated in Fig. 6, which comprises of $(3(N-1)+30)\cdot 30$ weights and $2\cdot 30 + N - 1$ biases. For a network with $N = 16$ nodes DASA has $2.5\cdot 10^3$ parameters. This number of parameters is several orders of magnitude smaller than the number of parameters employed by DNNs in typical deep learning tasks. For example, computer vision models specifically designed to be low-complexity, such as the MobileNetV2 architecture, employ over $3\cdot 10^6$ parameters, with competing vision models such as ResNet50 and InceptionV3 employing over $20\cdot 10^6$ parameters [39]. Our DASA implementation corresponds to fewer than $2.5\cdot 10^3$ products on inference – a computational burden which is feasible on real-time modern micro-controllers, and can support parallelization implemented by dedicated DNN hardware accelerators [40].

In online training utilized in multi-agent DRL, data is acquired by repeated interactions between distributed agents and the environment. In contrast, our proposed method enables nodes to learn the optimal synchronization parameters from a *single sequence of transmitted pulses*, and the trained DNNs can be subsequently employed at the respective nodes to maintain full clock (frequency and phase) synchronization between the nodes in the network. Thus, we avoid the repeated interactions utilized in DRL, which in the context of clock synchronization would imply repeated exchanges of N_T signals among the nodes, and would have decreased network's throughput. Nonetheless, in a dynamic network scenarios with highly mobile nodes, it is likely that the nodes may need to retrain their local models whenever the topology changes considerably from the one used during their training. Training schemes designed to facilitate online re-training in rapidly time-varying environments could leverage data from past topologies to predict future variations as in [41], [42]; however, we leave these extensions of DASA for future work.

V. Performance Evaluation

In this section we report an extensive simulation study to evaluate the performance of the proposed algorithm,[1] schematically described in Figs. 5 and 6. To facilitate a fair comparison between the DASA and the classic algorithm (4), the same parameters P_t, T_i, and $\{\phi_i(0)\}_{i=1}^N$, as well as the same topology, are used in the evaluation of both the analytic algorithm (4) and DASA. We recall that it was demonstrated in Section III that the analytic algorithm fails to achieve full synchronization for the considered scenario. DASA consists of three steps: 1) Data acquisition step; 2) Training step; and 3) Free-run or testing step. At the free-run step, the nodes use their trained DNNs to update their clocks via the update rule (7) with their measured $P_{i,j}(k)$'s and $t_{i,j}(k)$'s.

At startup, corresponding to clock index $k = 0$, each node i, $i \in \mathcal{I}_N$, sets its initial clock value $\phi_i(0)$ randomly and uniformly over $[0, T_i]$ (see Section III), and the DNN parameters $\boldsymbol{\theta}_i$ are initialized randomly according to the PyTorch default setting. Subsequently, the data acquisition step is applied at all nodes simultaneously. In this step, the nodes compute their clock times for pulse transmissions according to the update rule (7), where the outputs of the local DNNs at the nodes are computed

[1] The complete source code is available at https://github.com/EmekaGdswill/Distributed-DNN-aided-time-synchronization.git

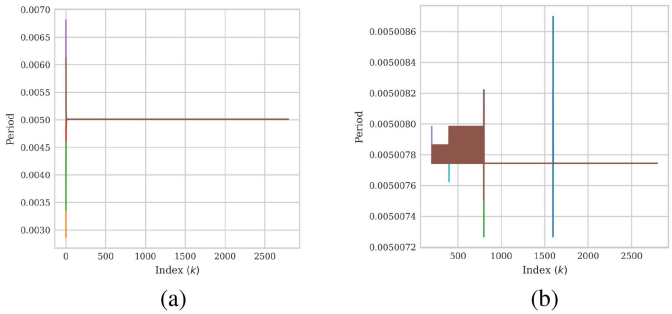


Fig. 7. Evolution of the clock periods for all 16 nodes using the proposed DASA: (a) Entire test period (time indices 0–2799); (b) Zoom on time indices 200–2799.

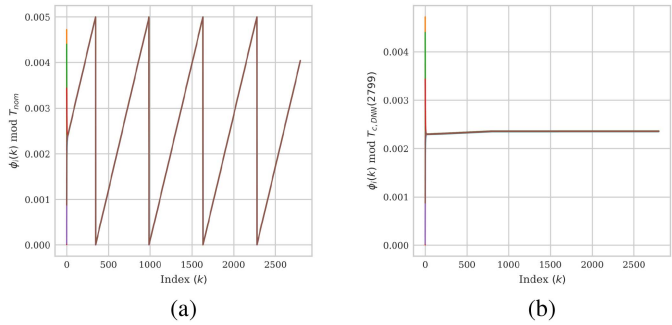


Fig. 8. Modulus of clock times $\phi_i(k)$ for all $N = 16$ clocks versus index k using the proposed DASA, (a) w.r.t T_{nom}; (b) w.r.t $T_{c,DNN}(2799)$.

with the corresponding *randomly initialized parameters*, $\boldsymbol{\theta}_i$, $i \in \mathcal{I}_N$, *which remain fixed* during the this step. We set the duration of the data acquisition interval to $N_T = 10$ TDMA cycles. At the end of the data acquisition interval, each node i has obtained a training data set $\mathcal{D}_i$. Next, the training step in applied at each node individually, where node i uses the data set $\mathcal{D}_i$ to train its individual DNN parameters $\boldsymbol{\theta}_i$, according to Algorithm 1. It is emphasized that the training process can be carried out concurrently with the data acquisition at the individual nodes, as each node applies processing based only on *its received pulse timings and powers*. We apply training over E epochs, determined such that the individual loss per node $\mathcal{L}_{\mathcal{D}_i}(\boldsymbol{\theta}_i)$, defined in (9), reaches the asymptotic value. After learning $\boldsymbol{\theta}_i$ is completed at node i, the node continues to update its clock using the rule (7), where the weighting of $\Delta\phi_{i,j}(k)$ is obtained by applying the trained DNN to the node's input data, i.e., computing $\psi_{\boldsymbol{\theta}_i}(\{(t_{i,j}(k) - \phi_i(k), P_{i,j}(k))\}_{j=1,j\neq i}^{N})$, where outputs corresponding to nodes whose transmitted signals were not received at node i are multiplied by zero. In the evaluations, we apply the testing step for 2800 time indexes.

From the numerical evaluations we identified that setting $E = 400$ epochs is sufficient for securing convergence. Recalling that each epoch corresponds to a single iteration, it is concluded that *convergence is relatively fast*. Consider first the convergence of the clock periods after training for the same network topology with $N = 16$ nodes, used in Section III (recall Fig. 3). Fig. 7 depicts the frequencies for all 16 nodes. From the time evolution in Fig. 7(a) it is observed that the nodes' period convergence is indeed rapid. Fig. 7(b) focuses on the last 2600 clock indexes of the testing step: Observe that after convergence, there are still small jumps in the period, which are much smaller than the mean value of the converged period, i.e., 6 orders of magnitude smaller, hence are considered negligible. We obtain that at the end of the testing step, the network attains a *mean synchronized period* of $T_{c,DNN}(2799) = 0.00500774$. Fig. 8(a) depicts the modulus of the clock times w.r.t T_{nom} across all the nodes, and Fig. 8(b) depicts the modulus of the clock times w.r.t $T_{c,DNN}(2799)$ across all the nodes. It is evident from the figure that the DNN-aided network attains full synchronization w.r.t. $T_{c,DNN}(2799)$, which is different from T_{nom}. Comparing Fig. 4(a) with Fig. 8(b) we conclude that the *proposed DASA offers significantly better performance than*

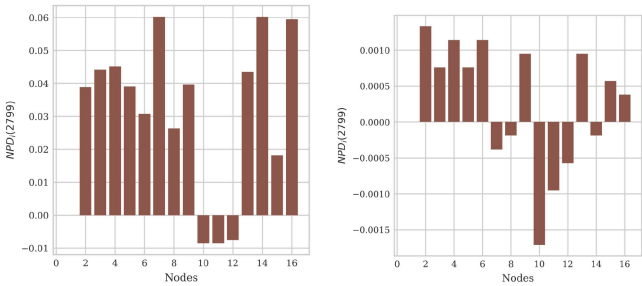


Fig. 9. NPD profile for (a) The classic algorithm; (b) DASA.

TABLE I
PERFORMANCE SUMMARY FOR THE CLASSIC ALGORITHM AND FOR THE PROPOSED DASA WITH ONLINE TRAINING, AT $k = 2799$

	classic algorithm	DASA (online training)
Mean period	0.00500208	0.00500774
STD of the period	$5.4715e^{-7}$	$< 10^{-10}$
Mean NPD	$3.0052e^{-2}$	$2.4995e^{-4}$
STD of NPD	$2.3738e^{-2}$	$8.3749e^{-4}$

the classic approach. Moreover, DASA's performance achieved using the trained DNN is *robust* to clock period differences and propagation delays.

We further compare the performance of both schemes by observing the normalized phase difference (NPD) at time k, defined as the difference between the clock phases at the nodes and the clock phase at node 1, normalized to the instantaneous mean period, denoted $T_c(k)$. Thus, the NPD for node i at time k is defined as:

$$\text{NPD}_i(k) = \left(\phi_i(k) - \phi_1(k)\right)/T_c(k), \tag{10}$$

$$\text{NPD range}(k) \triangleq \max_{i\in\mathcal{I}_N} NPD_i(k) - \min_{i\in\mathcal{I}_N} NPD_i(k), \tag{11}$$

where $T_c(k)$ depends on the tested algorithm: For the classic algorithm, the NPD is computed w.r.t. $T_c(k) \equiv T_{c,ANA}(k)$, and for the DASA the NPD is computed w.r.t $T_c(k) \equiv T_{c,DNN}(k)$. The NPD values for both schemes at $k = 2799$ are depicted in Fig. 9, and the mean and standard deviation (STD) of $\text{NPD}_i(k)$ over all $i \in \mathcal{I}_N$ at $k = 2799$ are summarized in Table I. From Fig. 9(a) it is observed that the NPD of the analytic algorithm spans a range of 7% of the clock period, with a mean NPD value

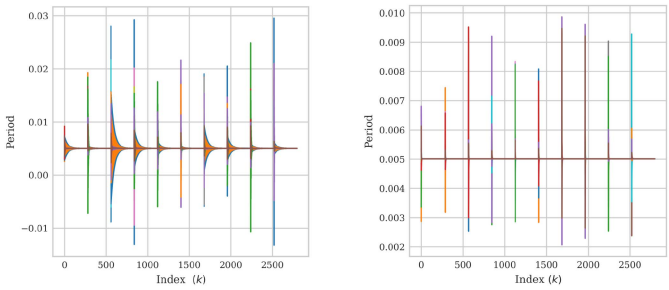


Fig. 10. Convergence of clock frequencies after clock phase and frequency resets over time index for (a) The classic algorithm; (b) DASA.

of 3%, while the DASA, depicted in Fig. 9(b), achieves an NPD range of 0.35% and a mean NPD of 0.025%. It thus follows that the DASA achieves an improvement by a factor of 28 in the standard deviation of the NPD and by a factor of 150 in the mean NPD. We observe from the table that both schemes achieve frequency synchronization, yet only the DNN-aided network achieves full and accurate synchronization. In the subsequent simulations we test the robustness of the DASA to initial clock phase and clock frequency values, and to node mobility, as well as characterize the performance attained when training is done offline.

A. Robustness to Clock Phase and Frequency Resets

We now test the robustness of DASA to clock frequency and phase resets during the free-run operation. Such variations may occur, e.g., due to clock temperature changes, hence the practical importance of this test. In the experiments, we first let the nodes learn their DNN networks' parameters, $\boldsymbol{\theta}_i$, $i \in \mathcal{I}_N$, in an unsupervised manner, as described in Algoithm 1. Then, DNNs' parameters at the nodes remain fixed, while clock resets are applied to 30% of the nodes every 280 time instants, where at each reset, the node's clock time and frequency were randomly generated according to the statistics detailed in Section III. Performance in terms of both speed of convergence after a clock reset and the ability to restore full network clock synchronization after a reset are presented for both DASA and the classic algorithm in Figs. 10 and 11, respectively, for the classic algorithm as well as for DASA. It is observed from Fig. 10 that both the classic algorithm and DASA are able to restore frequency synchronization, yet the proposed DASA is able to *instantly* restore frequency synchronization. The impact of the longer synchronization time of the classic algorithm is evident in Fig. 11, which shows that the slow frequency synchronization of the classic algorithm induces slow phase synchronization, which is not completed before the next reset occurs, while the newly proposed DASA instantly restores phase synchronization. It is observed in Fig. 11 that the converged (i.e. steady state) phases of DASA after clock resets are different, yet we clarify that this has no impact on communications network's performance as all the nodes converge to the *same phase* within the converged period. It is concluded that our proposed DASA is able to *instantly restore full synchronization*, while the classic algorithm requires considerably longer convergence times.

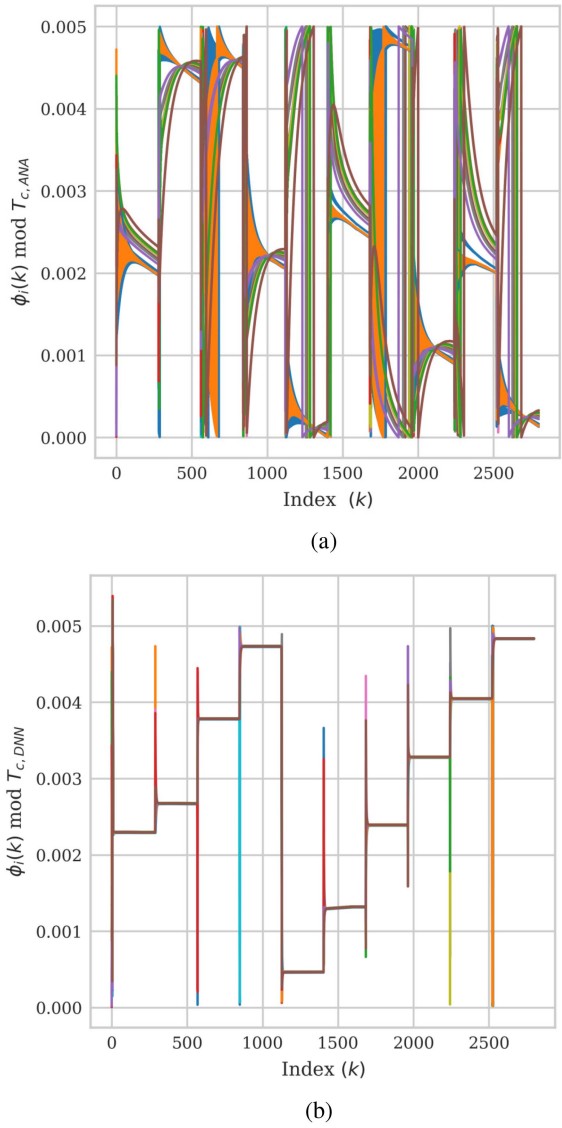


Fig. 11. Modulus of clock times after clock phase and frequency resets over time index for (a) The classic algorithm; (b) DASA.

Examine next the NPD maintained by DASA during the clock resets. To that aim we plot in Fig. 12 the NPD range, achieved by DASA when both clock phase and frequency resets are applied. Fig. 12(a) depicts the NPD over its entire range of NPD values, where a closeup on the range of smaller values, corresponding to the converged state is depicted in Fig. 12(b). Comparing Figs. 12(b) and 9(b) it is observed that after clock resets DASA achieves the *same NPD* as in the initial operation after training, namely, *clock resets do not increase the small NPD range achieved by DASA*. The experiments demonstrate that the proposed DASA is able to facilitate nearly uninterrupted clock phase synchronization, also in presence of random clock resets. These experiments clearly show that *DASA is robust to the values of the initial phases and the initial frequencies, and has*

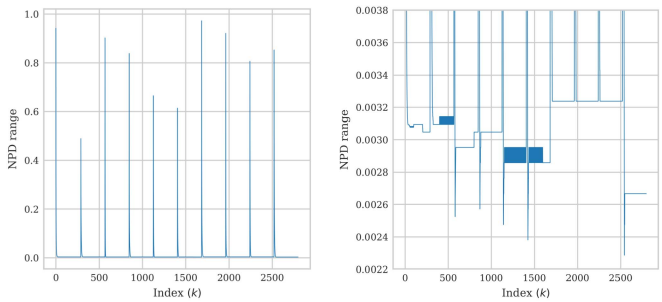


Fig. 12. NPD range over the time index with clock phase and frequency resets (a) Overall; (b) Closeup.

an outstanding ability to recover from variations in the phases and frequencies of the clocks.

B. Testing DASA Synchronization Performance With Mobile Nodes

In this subsection we test synchronization performance when some of the nodes are mobile. We let the DNNs at the nodes train for a stationary scenario (i.e., online training) and then test synchronization performance when a random subset of 30% of the nodes, selected uniformly, begins moving at a fixed speed, with each mobile is directed at some angular direction. Note that as the nodes move, the received signal powers from and at the moving nodes vary and the received signals at some of the nodes for certain links may drop below the detection threshold, which for the current setup is set to -114 [dBm]. This situation is implemented for such links by setting both the respective phase difference and the received power to zero. Naturally, this assignment should have a negative impact on synchronization accuracy. In the first experiment, in order to demonstrate the situation of node clustering, the moving nodes were all given the same direction of 95° and the moving speed was set such that at the end of the simulation each node has traversed 20 [Km]. Fig. 13 depicts the clock periods and clock phases modulo the instantaneous mean period, $T_{c,DNN}(k)$. It is observed from Fig. 13(a) that frequency synchronization is largely maintained also when nodes are mobile, yet, from Fig. 13(b) we observe a slow drift in the phase modulo $T_{c,DNN}(k)$, which implies that the period slightly varies as the nodes move. It is also noted that despite the phase drift, the nodes are able to maintain close phase values up to a certain time (in this simulation it is time index 1576, corresponding to a displacement of 10.9 [Km]), after which the phases split into two separate branches, one consisting of the five mobile nodes i.e., nodes 1, 2, 3, 11, and 12, and the second consisting of the stationary nodes. Checking the connectivity graph for this scenario, it was discovered that at this time index, the network splits into two disconnected sub-networks. Observe that at each sub-network the nodes maintain phase synchronization among themselves. Note that from Fig. 13 it follows that also in the presence of mobility, the performance of the DASA with online training is superior to that of the classic scheme without node mobility (cf. Figs. 13(b) and 4(a)). Therefore, it is not necessary to include here a mobility simulation for the classic algorithm.

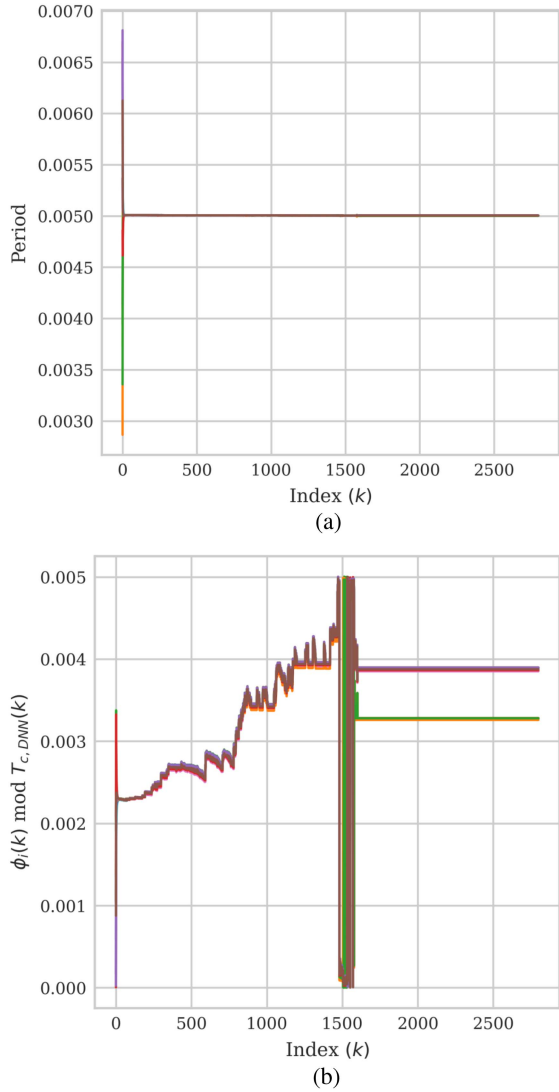


Fig. 13. Clocks' times and phases with mobility of 30% of the nodes at inference w.r.t. the time index: (a) Clock periods; (b) Clock times modulo the instantaneous period, $T_{c,DNN}(k)$.

Lastly, we take a closer look at the NPD performance before network splitting occurs. To that aim, we carried out a new set of 10 mobility experiments, where the mobile nodes are moving as a speed of 250 [Km/h] (here we may ignore the impact of the Doppler shift as it is less than 1 [ppm] for the relevant frequency band (e.g., around 900 [MHz]), which is much smaller than our considered frequency offsets), such that at the last time index, $k = 2799$, the distance each mobile node traversed was 1 [Km]. The direction of each moving node is randomly and uniformly selected over $[0, 2\pi)$. Fig. 14 depicts the evolution of the NPD range w.r.t $T_{c,DNN}(k)$, for all 10 experiments. It is observed that the NPD range maintains a very small value, and in fact, in all experiments the NPD range is increased by a factor smaller than 2 after a 1 [Km] displacement. Hence, we obtain that DASA exhibits a graceful degradation when the node

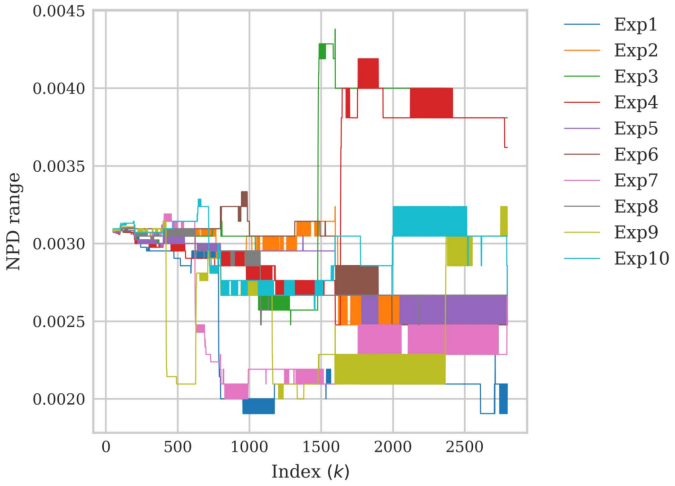


Fig. 14. NPD range for 10 mobility experiments, with online training, where each mobile node is moving at a different direction and traverses 1 [Km] during the simulation.

locations vary, and very good accuracy is maintained also for a large displacement of 1 [Km]. This demonstrates the robustness of DASA to topology variations of the network. This point is of major importance as it implies that moderate node displacements do not require retraining of the network. As a last note in this context, then, based on the mobility experiments and the clock reset experiments we conclude that the weights obtained by the training procedure (which minimizes the loss function (9)) are dependent on the relative locations of the nodes in the network and only weakly dependent on the clock frequencies and phases.

C. Performance of DASA With Offline Training

The offline training scheme of Section IV-C was tested using $N_{top} = 1000$ topologies, the mini-batch size was set to $N_{mb} = 10$; hence, there are $N_{batch} = N_{top}/N_{mb} = 100$ mini-batches. For the considered numerical evaluation with MB-SGD, setting $E = 3$ epochs was found sufficient to achieve convergence. After the DNNs have been trained over the set of $N_{top} = 1000$ network topologies, DASA was tested for new topologies not included in the training set. Fig. 15 depicts the results for a test topology sample: Fig. 15(a) demonstrates the rapid convergence of the clock periods to a mean synchronized period of $T_{c,DNN}(2799) = 0.00500679$. We observe some fluctuations in the periods of the nodes, however, the amplitudes of these variations are three orders of magnitude smaller than the mean synchronized period, hence, these variations are rather negligible. Fig. 15(b) depicts the modulus of the clocks' phases w.r.t. the mean synchronized period $T_{c,DNN}(2799)$. The figure demonstrates that *the proposed DASA with offline training achieves full clock synchronization*. Furthermore, its performance is significantly better than the performance of the classic scheme, as it is robust to propagation delays and clock frequency differences. Fig. 15(c), depicts a closeup of the NPD range. Observe that DASA achieves an NPD range of 0.41% at the first few time indices and a converged NPD range of 0.4% at later time indices ($k \geq 850$). Lastly, Fig 15(d) depicts a snapshot of the NPD

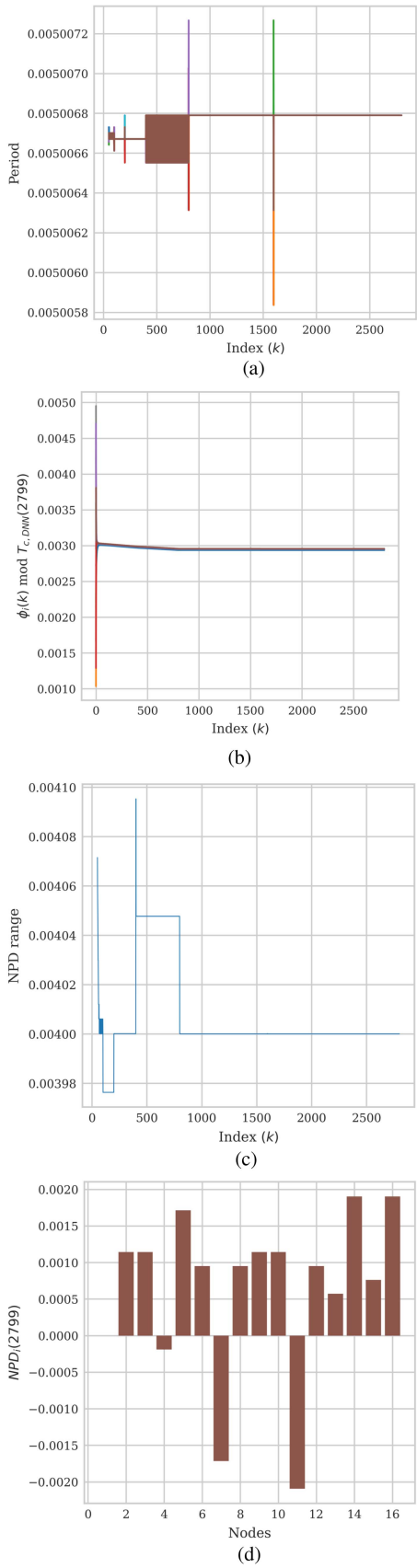


Fig. 15. Performance of the DASA for a test topology after offline training: (a) Clock periods; (b) Clock times modulo $T_{c,DNN}(2799)$; (c) NPD range; (d) $\{NPD_i(2799)\}_{i=1}^{16}$.

TABLE II
PERFORMANCE SUMMARY OF THE PROPOSED DASA WITH OFFLINE TRAINING, AT $k = 2799$ FOR A TEST NETWORK TOPOLOGY

	DASA (offline training)
Mean period	0.00500679
STD of the period	$< 10^{-10}$
Mean NPD	$6.4285e^{-4}$
STD of NPD	$1.1124e^{-3}$

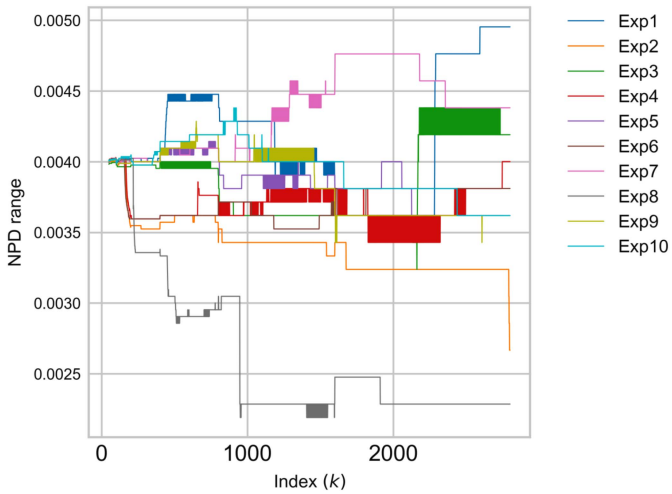


Fig. 16. NPD range for 10 experiments, with offline training, where each mobile node is moving at a different direction and traverses 1 [Km] during the simulation.

values across nodes at time $k = 2799$. From the figure, we again note that the NPD range is 0.4% across the nodes, we also see that the mean value is 0.064%. The performance of DASA for this test is summarized in Table II. Comparing with the online training results in Table I we note that period accuracy is similar for both scenarios; the main benefit of online training is a smaller mean NPD, by a factor of 2.5, and an NPD STD smaller by a factor of 1.3. Comparing the NPD of the offline DASA in Fig. 15(d) with that of the classic algorithm in Fig. 9(a) it is observed that the NPD range of the offline DASA is smaller than that of the classic scheme by an order of magnitude. Further comparing the mean NPD and the NPD STD of the two schemes via Tables I and II we conclude that the offline DASA improves the mean NPD by a factor of 50 and the NPD STD by a factor of approximately 20, compared to the classic scheme.

Lastly, we examine synchronization performance for the topology used in Fig. 15, with mobility as was done in Section V-B, i.e., we randomly and uniformly select 30% of the nodes, and for each selected nodes a random angular direction is selected uniformly over $[0, 2\pi)$. The moving nodes travel at a fixed speed, such that at the end of the simulation, each moving node has travelled 1 [Km]. Fig. 16 depicts the evolution of NPD range with time, as was done in Fig. 14 for online training. Specifically, it was observed that *at displacement of 1 [Km], The NPD range increases by a factor smaller than 2*, similar to online training. Thus, we conclude that *offline training also offers excellent robustness to node mobility.*

VI. CONCLUSION

We considered network clock synchronization for wireless networks via pulse-coupled PLLs at the nodes. The widely studied classic synchronization scheme based on the update rule (4) does not lead to full synchronization for networks with non-negligible propagation delays, and/or clock frequency differences between the nodes. In this work, we proposed an algorithm, abbreviated as DASA, which replaces the analytically computed weights of the classic scheme with weights generated using DNNs. A major novelty of this work is the design of an *unsupervised and distributed online learning scheme*, which requires *a very short training period* and has a low implementation complexity. These properties make the proposed algorithm very attractive for practical implementation. We also proposed an offline training algorithm which is based on *simulated data*. Numerical results show that when there are propagation delays and clock frequency differences between the nodes, *the proposed DASA is able to attain full synchronization* with a very high accuracy while the classic rule is not. It was demonstrated that DASA maintains synchronization also when clock frequency and phase resets occur at the nodes, and when some of the nodes are mobile. Lastly, with offline training, DASA was shown to achieve full synchronization, with only a small degradation in accuracy compared to online training.

REFERENCES

[1] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized 'zero-queue' datacenter network," in *Proc. ACM SIGCOMM Conf.*, 2014, pp. 307–318.
[2] Y. Geng et al., "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *Proc. USENIX Symp. Networked Syst. Des. Implementation*, 2018, pp. 81–94.
[3] 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.
[4] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul./Aug. 2004.
[5] 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.
[6] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. ACM Conf. Embedded Networked Sensor Syst.*, 2004, pp. 39–49.
[7] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588–2008 (Revision of IEEE Standard 1588-2002), 2008.
[8] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," Internet Engineering Task Force (IETF), RFC 5905, Jun. 2010.
[9] *IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Standard 802.1AS-2011, pp. 1–292, 2011.
[10] R. Pigan and M. Metter, *Automating With PROFINET: Industrial Communication Based on Industrial Ethernet*. Hoboken, NJ, USA: Wiley, 2008.
[11] Y.-W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 5, pp. 1085–1099, May 2005.
[12] 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.
[13] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Model-based machine learning for communications," 2021, *arXiv:2101.04726*.
[14] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, "Model-based deep learning: On the intersection of deep learning and optimization," *IEEE Access*, vol. 10, pp. 115384–115398, 2022.

[15] O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP J Wireless Commun. Netw.*, vol. 2007, pp. 1–13, Jun. 2007.
[16] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 841–848.
[17] M. K. Maggs, S. G. O'Keefe, and D. V. Thiel, "Consensus clock synchronization for wireless sensor networks," *IEEE Sens. J.*, vol. 12, no. 6, pp. 2269–2277, Jun. 2012.
[18] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Trans. Automat. Control*, vol. 50, no. 2, pp. 169–182, Feb. 2005.
[19] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Automat. Control*, vol. 50, no. 5, pp. 655–661, May 2005.
[20] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.
[21] M. A. Alvarez and U. Spagnolini, "Distributed time and carrier frequency synchronization for dense wireless networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 683–696, Dec. 2018.
[22] J.-C. Guey, "The design and detection of signature sequences in time-frequency selective channel," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2008, pp. 1–5.
[23] N. Abedini, S. Tavildar, J. Li, and T. Richardson, "Distributed synchronization for device-to-device communications in an LTE network," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1547–1561, Feb. 2016.
[24] F. Tong and Y. Akaiwa, "Theoretical analysis of inter-basestation-synchronization system," in *Proc. IEEE Int. Conf. Universal Pers. Commun.*, 1995, pp. 878–882.
[25] E. Sourour and M. Nakagawa, "Mutual decentralized synchronization for intervehicle communications," *IEEE Trans. Veh. Technol.*, vol. 48, no. 6, pp. 2015–2027, Nov. 1999.
[26] D. T.-L. Roche, B. Champagne, I. Psaromiligkos, and B. Pelletier, "On the use of distributed synchronization in 5G device-to-device networks," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.
[27] K. D. Pham, "Stochastic power controls for distributed pulse-coupled synchronization," in *Proc. IEEE Aerosp. Conf.*, 2017, pp. 1–9.
[28] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.
[29] I. E. L. Hulede and H. M. Kwon, "Distributed network time synchronization: Social learning versus consensus," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 7, pp. 660–675, 2021.
[30] S. B. Amor, S. Affes, F. Bellili, U. Vilaipornsawai, L. Zhang, and P. Zhu, "Joint ML time and frequency synchronization for distributed MIMO-relay beamforming," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–8.
[31] D. O. Reudink, "Large-scale variations of the average signal," in *Microwave Mobile Communications*. W. C. Jakes, Ed. Hoboken, NJ, USA: Wiley, 1974.
[32] P. Palá-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. Regular Papers*, vol. 62, no. 3, pp. 791–798, Mar. 2015.
[33] N. Naik, "LPWAN technologies for IoT systems: Choice between ultra narrow band and spread spectrum," in *Proc. IEEE Int. Syst. Eng. Symp.*, 2018, pp. 1–8.
[34] A. C. Scogna et al., "RFI and receiver sensitivity analysis in mobile electronic devices," in *Proc. High-Speed Commun. Syst. Des. Conf.*, 2017, pp. 1–6.
[35] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *Proc. IEEE*, vol. 111, no. 5, pp. 465–499, May 2023.
[36] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010 .
[37] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, May 2019.
[38] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, Univ. Toronto, Toronto, ON, Canada, 2013.
[39] C. Luo, X. He, J. Zhan, L. Wang, W. Gao, and J. Dai, "Comparison and benchmarking of AI models and frameworks on mobile devices," 2020, *arXiv:2005.05085*.
[40] 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.
[41] T. Raviv et al., "Meta-ViterbiNet: Online meta-learned Viterbi equalization for non-stationary channels," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2021, pp. 1–6.
[42] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online meta-learning for hybrid model-based deep receivers," *IEEE Trans. Wireless Commun.*, early access, Feb. 8, 2023, doi: 10.1109/TWC.2023.3241841.

Emeka Abakasanga received the B.Sc. degree in communications engineering from the University of Science and Technology, Ghana, in 2014, and the M.Sc. degree in communications engineering from Loughborough University, Loughborough, U.K., in 2015. He is currently a Doctoral Researcher with the Network Information Laboratory, Ben-Gurion University, Beersheva, Israel. His research interests include information theory and signal processing for communications.

Nir Shlezinger (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from Ben-Gurion University, Israel, in 2011, 2013, and 2017, respectively. He is currently an Assistant Professor with the School of Electrical and Computer Engineering, Ben-Gurion University, Beersheva, Israel. From 2017 to 2019, he was a postdoctoral Researcher with the Technion, and from 2019 to 2020, he was a postdoctoral Researcher with the Weizmann Institute of Science, Rehovot, Israel, where he was awarded the FGS prize for outstanding research achievements. His research interests include communications, information theory, signal processing, and machine learning.

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