DOI: 10.1109/EuCNC48522.2020.9200930

Synchronization in 5G: a Bayesian Approach

Meysam Goodarzi,*† Darko Cvetkovski,*† Nebojsa Maletic,* Jesús Gutiérrez,* and Eckhard Grass,*†
*IHP – Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
†Humboldt University of Berlin, Berlin, Germany.
Emails:{goodarzi, cvetkovski, maletic, teran, grass}@ ihp-microelectronics.com

***Abstract*—In this work, we propose a hybrid approach to synchronize large scale networks. In particular, we draw on Kalman Filtering (KF) along with time-stamps generated by the Precision Time Protocol (PTP) for pairwise node synchronization. Furthermore, we investigate the merit of Factor Graphs (FGs) along with Belief Propagation (BP) algorithm in achieving high precision end-to-end network synchronization. Finally, we present the idea of dividing the large-scale network into local synchronization domains, for each of which a suitable synchronization algorithm is utilized. The simulation results indicate that, despite the simplifications in the hybrid approach, the error in the offset estimation remains below 5 ns.**

***Index Terms*—5G, Synchronization, Kalman Filtering, Factor Graph, Belief Propagation, Hybrid Synchronization**

I. Introduction

The fifth generation of wireless networks (5G) are expected to deliver a wide variety of services, many of which require the nodes' clocks to be aligned. Distributed beamforming [1], tracking and mobility prediction [2], [3], and localization [4] can be referred to as cases where time synchronization is necessary to guarantee certain levels of quality of service. There has been a great effort to design algorithms for achieving fast and continuous synchronization [5]. Generally, state-of-the-art sync[1] algorithms can be classified into two categories: a) pairwise synchronization [6]–[8], where protocols are primarily designed to synchronize two nodes, and b) network-wide synchronization, where protocols are designed to synchronize a large number of nodes in the network [9], [10].

Among all the existing synchronizers, perhaps IEEE 1588 [11], often denoted as Precision Time Protocol (PTP), is the most well-known sync protocol employed in a wide variety of applications. PTP along with the Best Master Clock Algorithm (BMCA) uses hardware time-stamping and pairwise communication between nodes to determine the Master Node (MN) and to perform synchronization. Although this combination might offer satisfactory performance in networks with medium time precision sensitivity, errors in time-stamping on one hand [6], and the BMCA failure in determining the MN on the other hand [12], can lead to a significant deterioration of the performance in time precision sensitive networks. The former results from the layer where the time-stamps are taken, while the latter can be potentially due to the fact that the communication network might be based on a mesh topology. In [6] and [8] the substantial benefit of Kalman Filtering (KF) has been revealed, whereby the negative impact of the time-stamping error on the sync processes is alleviated. Furthermore, [9] proposes a network-wide synchronization where Factor Graph (FG) is used along with Belief Propagation (BP) for the nodes to perform synchronization. Unlike BMCA, in BP the nodes exchange their opinion about each other, thereby reaching an agreement about their clock status even if the network (or its corresponding FG) contains loops.

While the above-mentioned works have made valuable contributions towards synchronization, it is highly unlikely that each individual solution, e.g. the ones in [6] and [9], can alone achieve the high precision aimed by 5G while keeping the complexity low. To bring both types of algorithms together, the idea of a *synchronization harmonizer* has been introduced in [13]. One of the key aspects thereof is to equip the network with different sync algorithms (or a combination thereof). In fact, in order for the harmonizer to meet the desirable sync precision, one can divide the large scale network into multiple local synchronization domains and employ the suitable sync algorithm based on each local network topology. In this manner, it is easier to satisfy the requirement of the relative time error in the sync domains, i.e. each can run the best suitable algorithm based on its topology and capabilities [14].

The contribution of this paper is summarized as follows:

- We analyze the statistical relation between the neighboring nodes with the aid of PTP time-stamp exchange.
- We discuss pairwise and network-wide statistical synchronization algorithms based on KF and FG, respectively.
- We propose a hybrid approach to achieve high precision time synchronization across the network.

The rest of this paper is structured as follows: In Section II, we introduce our system model and obtain the statistics between the nodes. In Section III, the clock offset estimation based on the obtained statistics is discussed. Furthermore, simulation results are presented and discussed in Section IV. Finally, Section V concludes this work and indicates the future work.

Notation: The boldface capital $\boldsymbol{A}$ and lower case $\boldsymbol{a}$ letters denote matrices and vectors, respectively. $\mathbf{1}_N$ is a vector with N entries each equal to 1. $\boldsymbol{I}_N$ is a $N \times N$ dimensional identity matrix. The symbol $\propto$ represents the linear scalar relationship between two real valued functions. $\mathrm{Var}(\cdot)$ and $\mathcal{E}\{\cdot\}$ denote the variance and statistical expectation, respectively. $\mathcal{N}(\mu, \sigma^2)$

The research leading to these results has received funding from the European Union's Framework Programme Horizon 2020 for research, technological development and demonstration under grant agreement No. 762057 (5G-PICTURE).

[1]The words "synchronization" and "sync" are used alternatively in this paper and carry the same meaning.

represents a Gaussian distribution with mean μ and variance σ^2.

II. System Model

A. Clock Model

Each node i is considered to have the clock model

$$c_i(t) = \gamma_i t + \theta_i \tag{1}$$

where γ_i and θ_i denote the clock skew and offset, respectively. Furthermore, t represents the reference time. In fact, function $c_i(t)$ determines how the reference time and clock of node i are mapped onto each other. Given that, the goal of time synchronization [2] is to find the offset θ_i (or a transformation thereof) for each node and apply corrections such that, ideally, all the clocks show the same time as the reference time.

B. Offset Decomposition

To achieve the above-mentioned goal, we begin with decomposing the offset between two nodes, thereby acquiring a reasonable conception of the offset components to be compensated. The offset θ_i is comprised of several components, as shown in Figure 1. t_A (and t_B) is the time that a packet needs to leave the transmitter after being time-stamped [3], d_{AB} and d_{BA} denote the propagation delay, and r_B (and r_A) is the time that a packet needs to reach the time-stamping point upon arrival at the receiver. In general, the packets sent from node A to node B do not experience the same delay as the packets sent from node B to node A. In other words

$$t_A + d_{AB} + r_B \neq t_B + d_{BA} + r_A.$$

Furthermore, we can define $T = t_A + r_B$, and $R = t_B + r_A$. Generally, T and R are random variables due to several independent random processes and therefore can be assumed i.i.d. Gaussian random variables, whereas d_{AB} and d_{BA} are due to propagation and usually assumed to be deterministic and symmetric ($d_{AB} = d_{BA}$)[9].

C. Measurement Model

We use the time-stamping shown in Figure 2, implemented by the PTP protocol [11], to estimate the offset between two adjacent nodes. Thus for the k-th round of message exchange we can write

$$\alpha_{ij}(c_i(t_2^k) - \theta_i) = c_j(t_1^k) - \theta_j + \gamma_j(d_{ij} + T_k), \tag{2}$$
$$\alpha_{ij}(c_i(t_3^k) - \theta_i) = c_j(t_4^k) - \theta_j - \gamma_j(d_{ij} + R_k), \tag{3}$$

where t_1^k/t_4^k and t_3^k/t_2^k are the time points where neighboring nodes i and j send/receive the sync messages, respectively. Moreover, $\alpha_{ij} = \frac{\gamma_j}{\gamma_i}$ represents the relative clock skew. Generally, the skew of a properly working clock is considered to be close to 1 [15]. In fact, given that the term $d_{ij} + T_k$ and

[2]In this work, we focus only on offset estimation and leave the skew estimation for the future works. In fact, the goal of this work is only to reveal the potential performance of hybrid synchronization.

[3]From now on, the term "time-stamp" refers to hardware time-stamping.

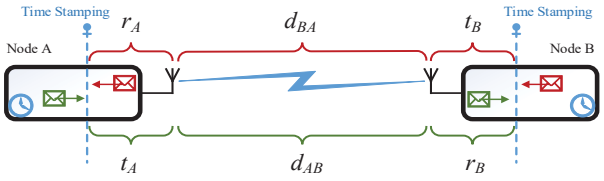

Fig. 1. Delay Decomposition.

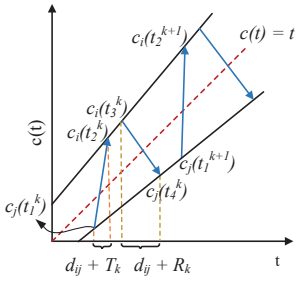

Fig. 2. Message exchange between adjacent nodes.

$d_{ij} + R_k$ are expected to be of low value, we can adopt the approximation in [2] given by

$$\gamma_j(d_{ij} + T_k) \approx (d_{ij} + T_k), \tag{4}$$
$$\gamma_j(d_{ij} + R_k) \approx (d_{ij} + R_k). \tag{5}$$

Consequently (2) and (3) turn into

$$\alpha_{ij}c_i(t_2^k) = c_j(t_1^k) - \theta_j + \alpha_{ij}\theta_i + d_{ij} + T_k, \tag{6}$$
$$\alpha_{ij}c_i(t_3^k) = c_j(t_4^k) - \theta_j + \alpha_{ij}\theta_i - d_{ij} - R_k. \tag{7}$$

Nevertheless $\alpha_{ij} = \frac{\gamma_j}{\gamma_i} \approx 1$ does not hold since the value of time-stamps $c_i(t_2^k)$ and $c_i(t_3^k)$ can be very large and therefore even the small amount of α_{ij} could lead to a considerable difference in their multiplication, and consequently in the estimation of the clock offsets. For the sake of simplicity we change the notation of the time-stamps, e.g., $c_j(t_1^k)$ is denoted by $c_{j,1}^k$. Summing up (6) and (7), we can write

$$\alpha_{ij}(c_{i,2}^k + c_{i,3}^k) = c_{j,1}^k + c_{j,4}^k - 2(\theta_j - \alpha_{ij}\theta_i) + Z_k, \tag{8}$$

where $\{Z_k = T_k - R_k\}_{k=1}^{k=K}$ are i.i.d. Gaussian random variables with mean $\mu = \mu_T - \mu_R$ and the variance $\sigma^2 = \sigma_T^2 + \sigma_R^2$. The mean, μ, is expected to be close to 0 and therefore negligible [9]. The parameter σ^2 is assumed to be static as it is mostly due to the hardware properties of the nodes [15] and can be computed by the nodes in a defined training phase. In the following, we utilize the properties of PTP time-stamping to estimate the parameters α_{ij} and σ^2.

1) Relative clock skew α_{ij}: Based on the Simple Skew Clock model introduced in [16] and employed in [6], [8], the individual skews, and consequently the corresponding relative skews are assumed to be constant within each sync period. A simple PTP-based estimation of α_{ij} for a single sync period can be given by

$$\alpha_{ij} = 1 + \mathcal{E}\{\zeta\}, \tag{9}$$

where

$$\zeta = \frac{\left(c_{i,2}^{k} - c_{i,2}^{k-1}\right) - \left(c_{j,1}^{k} - c_{j,1}^{k-1}\right)}{c_{j,1}^{k} - c_{j,1}^{k-1}}. \tag{10}$$

The parameter ζ denotes the clock drift between the two nodes. In fact, (9) states that the relative skew is equal to the mean of the relative skews in one set of PTP time-stamp exchange (a single set can include K rounds of time-stamp exchange).

2) Variance σ^2: Subtracting (7) from (6) gives

$$\alpha_{ij}(c_{i,2}^{k} - c_{i,3}^{k}) - (c_{j,1}^{k} - c_{j,4}^{k}) = 2d_{ij} + T_k + R_k. \tag{11}$$

σ^2 is calculated in the first communication of two nodes in the course of a training phase. In particular, repeating the message exchange during this phase provides the nodes with sufficient samples to calculate the variance of $2d_{ij} + T_k + R_k$ which is $\sigma^2 = \sigma_T^2 + \sigma_R^2$. It is clear that $\mathrm{Var}(T_k + R_k) = \mathrm{Var}(T_k - R_k)$ since T_k and R_k are independent.

D. Pairwise Conditional Probability

The aim here is to define the conditional probability of delay between two adjacent clocks given their offsets, θ_i and θ_j. Given that Z_k is Gaussian distributed, (8) states that the relation between the set of time-stamps, $\mathbf{c}_{i\to j}$ and $\mathbf{c}_{j\to i}$, and the offset parameters, θ_i and θ_j, is as follows:

$$P(\mathbf{c}_{ij}|\theta_i, \theta_j) = \tag{12}$$

$$\left(\frac{1}{\sqrt{2\pi}\alpha_{ij}\sigma}\right)^{K} \exp\left(-\frac{\| \mathbf{c}_{ij} - 2(\theta_j - \alpha_{ij}\theta_i)\cdot \mathbf{1}_K \|^2}{2\alpha_{ij}^2\sigma^2}\right) \tag{13}$$

$$= \left(\frac{1}{\sqrt{2\pi}\alpha_{ij}\sigma}\right)^{K} \exp\left(-\frac{4K}{2\alpha_{ij}^2\sigma^2}\left[(\alpha_{ij}\theta_i - \theta_j) + \frac{1}{2K}\mathbf{1}_K^T\mathbf{c}_{ij}\right]^2\right), \tag{14}$$

where

$$\mathbf{c}_{ij} = \left[c_{ij}^1, \cdots, c_{ij}^K\right] = \alpha_{ij}\mathbf{c}_{j\to i} - \mathbf{c}_{i\to j},$$

and

$$\mathbf{c}_{j\to i} = \left[c_{i,2}^1 + c_{i,3}^1, \cdots, c_{i,2}^K + c_{i,3}^K\right],$$
$$\mathbf{c}_{i\to j} = \left[c_{j,1}^1 + c_{i,4}^1, \cdots, c_{i,1}^K + c_{i,4}^K\right].$$

With above conditional probability distribution defined, the Bayesian posterior distribution of clock offset, θ_i, is given by

$$p(\theta_i|\mathbf{c}_{ij}) = \int p(\theta_i, \theta_j|\mathbf{c}_{ij})d\theta_j \propto \int p(\mathbf{c}_{ij}|\theta_i, \theta_j)p(\theta_i)p(\theta_j)d\theta_j. \tag{15}$$

where $p(\theta_i)$ and $p(\theta_j)$ denote the prior distribution of θ_i and θ_j, respectively, and assumed to be Gaussian [2]. Consequently, the clock offset for node i can be estimated as

$$\hat{\theta}_i = \arg\max_{\theta_i} p(\theta_i|\mathbf{c}_{ij}) = \arg\max_{\theta_i} \int p(\theta_i, \theta_j|\mathbf{c}_{ij})d\theta_j. \tag{16}$$

In the next section, we firstly estimate the clock offsets for the pairwise synchronization. Later on, we extend (15) and (16) for the network-wide synchronization where the offsets are estimated considering the impact of all nodes on each other.

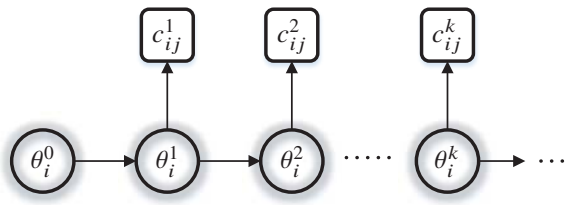

Fig. 3. Bayesian representation of offset estimation.

III. Clock Offset Estimation

A. Pairwise KF-based Synchronization

Let us assume that the state of θ_i after k-th round of time-stamp exchange is denoted by θ_i^k (Figure 3). The probability distribution function (pdf) corresponding to k-th state can then be written as

$$p(\theta_i^k|\mathbf{c}_{ij}) = \int p(\theta_i^0, \cdots, \theta_i^k|\mathbf{c}_{ij})\ d\Theta^{k-1}, \tag{17}$$

where $\Theta^{k-1} = \left[\theta_i^0, \cdots, \theta_i^{k-1}\right]$. Employing Bayes rule:

$$p(\theta_i^k|\mathbf{c}_{ij}) \propto \int p(\mathbf{c}_{ij}|\theta_i^0, \cdots, \theta_i^k)p(\theta_i^0, \cdots, \theta_i^k)\ d\Theta^{k-1}. \tag{18}$$

Assuming independent measurements and the Markov property, the terms in the integral can be rewritten as

$$\begin{aligned} p(\mathbf{c}_{ij}|\theta_i^0, \cdots, \theta_i^k) &= p(c_{ij}^1|\theta_i^1)\cdots p(c_{ij}^k|\theta_i^k), \\ p(\theta_i^0, \cdots, \theta_i^k) &= p(\theta_i^k|\theta_i^{k-1})\cdots p(\theta_i^1|\theta_i^0)p(\theta_i^0). \end{aligned} \tag{19}$$

Plugging (19) into (18) leads to

$$p(\theta_i^k|\mathbf{c}_{ij}) \propto \underbrace{\int p(\theta_i^0)\left[\prod_{r=1}^{k-1} p(\theta_i^r|\theta_i^{r-1})p(c_{ij}^r|\theta_i^r)\right] p(\theta_i^k|\theta_i^{k-1})d\Theta^{k-1}}_{p(\theta_i^k|c_{ij}^{1:k-1})}\ p(c_{ij}^k|\theta_i^k), \tag{20}$$

which can be simplified as follows:

$$p(\theta_i^k|\mathbf{c}_{ij}) \propto p(\theta_i^k|c_{ij}^{1:k-1})p(c_{ij}^k|\theta_i^k). \tag{21}$$

Assuming Gaussian distribution for the conditional probabilities in (21), the KF equations can readily be derived [17].

In practice, there is always uncertainty in time-stamping, i.e. the time a packet is stamped is different from the time it actually leaves/enters a transmitter/receiver. Given that, it appears necessary to preprocess the measured quantities (offset and drift) in order to obtain their true values. While the estimation in equation (21) can be directly used to preprocess the measured offset and drift [10], for the sake of clarity and tractability, we present the KF equations corresponding to (21) which can then be employed to make as precise an estimation as possible [6].

We begin with the first KF equation, typically known as prediction equation. It is given by [8]

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \omega_k. \tag{22}$$

Bringing (22) in the context of pairwise synchronization and assuming the process noise is negligible ($\omega_k = 0$)

$$\begin{bmatrix} \theta_i^k \\ \zeta_i^k \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_i^{k-1} \\ \zeta_i^{k-1} \end{bmatrix}, \tag{23}$$

where ΔT denotes the time needed for one round of time-stamp exchange. Furthermore, the measurement vector is

$$\mathbf{z} = \begin{bmatrix} \tilde{\theta}_i^k \\ \tilde{\zeta}_i^k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_i^k \\ \zeta_i^k \end{bmatrix} + \begin{bmatrix} \kappa_\theta \\ \kappa_\zeta \end{bmatrix}. \tag{24}$$

The parameters κ_θ and κ_ζ represent the noise in the measurement of θ_i^k and ζ_i^k, respectively. The measured value $\tilde{\zeta}_i^k$ can be obtained using (10) while the measured offset is calculated by [6]

$$\tilde{\theta}_i^k = \frac{1}{2}\left[\left(c_{i,2}^k + c_{i,3}^k\right) - \left(c_{j,1}^k + c_{j,4}^k\right)\right]. \tag{25}$$

The prediction equations can then be rewritten as

$$\mathbf{x}_{k|k-1} = \mathbf{A}\mathbf{x}_{k-1}, \tag{26}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T, \tag{27}$$

where $\mathbf{P}_{k-1}$ and $\mathbf{P}_{k|k-1}$ denote a prior and a posterior prediction covariance matrix, respectively. Moreover, we can write the update equations as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\left[\mathbf{P}_{k|k-1} + \mathbf{R}\right]^{-1}, \tag{28}$$

$$\mathbf{x}_k = \mathbf{x}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z} - \mathbf{x}_{k|k-1}\right), \tag{29}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\right)\mathbf{P}_{k|k-1}, \tag{30}$$

where $\mathbf{K}_k$ is the Kalman gain and $\mathbf{R}$ denotes the measurement noise covariance matrix and can be given by [6]

$$\mathbf{R} = \sigma^2 \begin{bmatrix} 1 & \frac{1}{\Delta T} \\ \frac{1}{\Delta T} & \frac{2}{(\Delta T)^2} \end{bmatrix}. \tag{31}$$

The parameter σ^2 is calculated as explained in section II-C2.

B. Network-wide FG-based Synchronization

1) Introduction: FGs are used to represent the factorization of pdfs. As shown in Figure 4 (the green graph), a FG comprises a number of nodes, each denoted by a variable and several factor nodes, each being a function of their neighboring variables. In particular, the factorization and graph structure in FGs can preserve the information about the form of the distribution while alleviating the computation load, e.g. that of marginal distribution through the sum-product algorithm [18]. Extending (15) to the whole network, the Bayesian posterior distribution of each node θ_i can be written as

$$p(\theta_i|\{\mathbf{c}_{ij}\}_{i=1:M,j\in ne(i)}) = \int p(\theta_1,\cdots,\theta_M|\{\mathbf{c}_{ij}\}_{i=1:M,j\in ne(i)})d\theta_1\cdots d\theta_{i-1}\, d\theta_{i+1}\cdots d\theta_M, \tag{32}$$

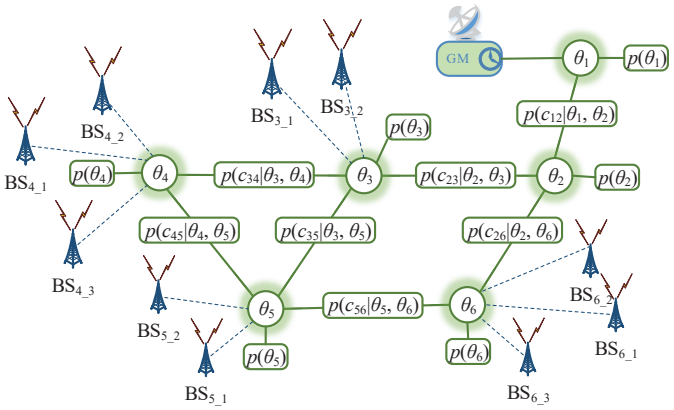

Fig. 4. Factor graph corresponding to an exemplary communication network.

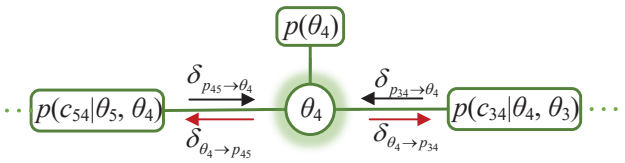

Fig. 5. Message passing in Belief Propagation.

where

$$p(\theta_1,\cdots,\theta_M|\{\mathbf{c}_{ij}\}_{i=1:M,j\in ne(i)}) \propto \prod p(\theta_i)\prod p(\mathbf{c}_{ij}|\theta_i,\theta_j), \tag{33}$$

$ne(i)$ represents the set of neighboring nodes of node i, and M denotes the total number of nodes in the network. In general, the computation of the marginal in (32) is costly and of NP-hard complexity. However, there are several methods in the literature to efficiently represent the joint probability and compute the exact or approximate marginals. BP is an algorithm which can run on FG and relies on exchanging beliefs between neighboring nodes to compute the marginals. In the following, we briefly introduce the BP algorithm.

2) Belief Propagation: Figure 5 depicts the principles of message passing in the BP algorithm for the exemplary node θ_4. For the sake of simplicity, we denote the factor $p(\mathbf{c}_{ij}|\theta_i,\theta_j)$ by p_{ij}. The message from a variable vertex θ_i to factor vertex p_{ij} is given by [19]

$$\delta_{\theta_i\to p_{ij}}(\theta_i) = p(\theta_i)\prod_{p_{ik}\in\{ne(\theta_i)\setminus p_{ij}\}}\delta_{p_{ik}\to\theta_i}(\theta_i), \tag{34}$$

where $\delta_{p_{ik}\to\theta_i}(\theta_i)$ is the message from a factor vertex p_{ik} to the variable vertex θ_i, and is given by

$$\delta_{p_{ij}\to\theta_i}(\theta_i) = \int p(\mathbf{c}_{ij}|\theta_i,\theta_j)\delta_{\theta_j\to p_{ij}}(\theta_j)d\theta_j. \tag{35}$$

It is straightforward to see that

$$b(\theta_i) \propto p(\theta_i)\prod_{p_{ik}\in ne(\theta_i)}\delta_{p_{ik}\to\theta_i}(\theta_i), \tag{36}$$

where $b(\theta_i)$ denotes the marginal belief of variable node θ_i. The BP procedure can be summarized as

1) The message $\delta_{\theta_i \to p_{ij}}(\theta_i)$ is transmitted from θ_i to the neighboring factor node p_{ij} (it is initialized non-informatively in the first iteration),
2) The factor node p_{ij} compute the message $\delta_{p_{ij} \to \theta_i}(\theta_i)$ based on its incoming messages and send the calculated message to the neighboring node θ_i,
3) Each node i updates its belief $b(\theta_i)$ based on the received messages from the neighboring factor nodes.

We note that in practice there are neither factors nor variable nodes meaning that (34) and (35) are calculated locally at each node and only $\delta_{p_{ik} \to \theta_i}(\theta_i)$ is sent to the neighboring node i.

C. Hybrid BP-KF

Given Sections III-A and III-B, one can decide on the suitable algorithm for each sync domain in network. That is, the nodes backhauling the BSs need to be precisely synchronized using BP whereas the nodes at the edge of the network (BSs) can be synchronized using KF where mostly global sync is of less importance compared to local sync precision.

Algorithm 1 describes the steps of the hybrid synchronization approach. Firstly, in step 1, we decide on the network sections where BP and KF are to be applied (they are labeled as BP-nodes and KF-nodes, respectively). Later, in step 2, the time-stamp exchange mechanism shown in Figure 2 and, correspondingly, the KF algorithm is initiated at the KF-nodes. In step 3, the time-stamp exchange is initiated among the BP-nodes, thereby obtaining the required time-stamps to calculate the conditional probability in (14). The BP iterations begin at step 4 and continue until convergence or when the maximum number of iterations L is reached. In step 5, each BP-node calculates its outgoing messages and sends them to their corresponding nodes. Each node's belief is then computed in step 6 using (36). Steps 7-9 are responsible to check the convergence by comparing the difference of clock offset estimations in iterations (l) and $(l-1)$ with a predefined small value ϵ. It is noteworthy that the step 2 and steps 3-10 can continuously run in parallel.

IV. Simulation Results

We consider the network in Figure 4 as an exemplary scenario, where a number of BSs are backhauled by a mesh network. We conduct two sets of simulations: a) synchronizing the whole network based only on FG and, correspondingly, BP algorithm (the BSs in Figure 4 are assumed to be variable nodes as well and connected to the mesh network via factors), and b) we perform synchronization in a hybrid manner where the mesh backhauling network is synchronized based on FG while the BSs at the edge of the network are being synchronized using KF. We then compute Root Mean Square Error (RMSE) of offset estimation as a measure to evaluate the performance in each scenario. For the sake of simplicity and without loss of generality we consider only the nodes θ_4 and θ_6 and their corresponding BSs. Moreover, the simulation parameters are set as in Table I and the Python package in [20] is employed to perform the message passing algorithm.

Algorithm 1 Network synchronization algorithm

1: Determine the suitable algorithm for each part of the network (BP-nodes or KF-nodes).
2: Start the time-stamping exchange and correspondingly the KF algorithm at KF-nodes.
3: Start the time-stamp exchange between adjacent BP-nodes and Calculate (14) for each pair
4: **for** $l = 1, 2, \cdots, L$ **do**
5: Compute the messages using (34) and (35) for each BP-node and transmit them to its neighboring nodes
6: Compute the offset estimation at each BP-node using (36) and update their belief
7: **if** $\hat{\theta}_i^{(l)} - \hat{\theta}_i^{(l-1)} \leq \epsilon \ \forall i$ **then**
8: Go to step 3
9: **end if**
10: **end for**

TABLE I
Simulation Parameters

Number of independent simulations	10000
Initial random delays	[-50, 50] ns
Number of time-stamp exchange K	10
Standard deviation of T_k and R_k	4 ns
Random delay between each pair of nodes	[200, 300] ns
Initial pdf of the offset for each node	$\mathcal{N}(0, +\infty)$
Initial pdf of the offset of MN	$\mathcal{N}(0, 0)$

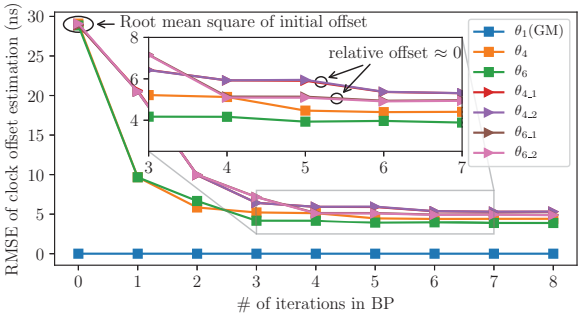

Fig. 6. BP applied on the whole network (scenario a).

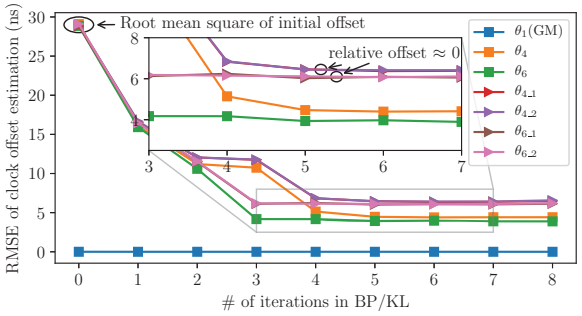

Fig. 7. BP applied only to the backhauling mesh network (scenario b).

Figure 6 represents RMSE of offset estimation for scenario (a). As can be seen, the BP converges after 4 iterations achieving a synchronization accuracy of 4−6 ns with respect to the Grand Master (GM) node. The convergence is guaranteed

for networks with at least one GM [9]. However, when a network contains loops, the value to which BP converges, is considered to be approximate [19]. In fact, the results in this simulation setup reveal the potential performance of BP for time synchronization in communication networks. However, the nodes, and particularly the BSs, must wait at least 4 iterations (or n iterations if there are n nodes between a BS and GM) to be completely synchronized. This can be problematic in a number of applications, e.g., localization, where continuous time alignment is essential. Therefore, it appears necessary for the BSs to synchronize themselves more frequently to be able to deliver certain services.

Figure 7 shows the RMSE of offset estimation for scenario (b). As can be observed, the performance slightly deteriorates (around 2 ns) compared to scenario (a). However, we note that the iterations of KF are significantly faster than that of BP. In fact, BP begins only when the nodes have already conducted several rounds of time-stamp exchange (in order to obtain the conditional probabilities) and, even then, it still needs 4 iterations to perform synchronization. In contrast, KF updates the estimation after each round of time-stamp exchange, thereby maintaining the relative clock offsets low. In other words, since the KF is faster and runs independently (does not need any information from the other network sections as BP does), it is able to conduct more iterations, thereby continuously fulfilling the local requirement of relative time error.

In summary, the simulation results indicate that BP can be of great potential for high precision network synchronization. Nevertheless, despite the excellent performance, high number of message passing iterations can cause trouble by prolonging the sync period. In particular, the time needed for the nodes to exchange time-stamps and pass messages can lead to deterioration in accuracy of synchronization. As a solution, the hybrid approach explained in section III-C can be adopted to alleviate the above-mentioned problem. That is, applying BP only on the critical parts of the network (e.g., the backhauling part which is responsible for distributing the clock to the edges) to achieve as high a precision as possible in global level. Moreover, faster algorithms, e.g. KF, can be readily employed on the edges of the network where precise, fast, and frequent local synchronization is required for numerous applications

V. Conclusion and Future Work

We presented two algorithms to synchronize the nodes in communication networks, each extensively discussed and shown to have benefits and drawbacks. One is based on Factor Graphs and able to achieve extremely accurate synchronization with higher complexity (high number of time-stamp exchanges and message passing iterations), while the other can deliver strong performance in tree structure networks. Further on, we combined the two approaches to maintain synchronization accuracy on a global level while performing frequent precise synchronization at local level. Simulation results show that the proposed hybrid network can achieve high precision and frequent synchronization at the cost of a slight deterioration in performance.

We only dealt with clock offset estimation, however skew compensation cannot be ignored while designing a sustainable synchronization algorithm. The future works aim at incorporating skew synchronization into the proposed algorithm to further enhance the performance.

References

[1] S. Jagannathan, H. Aghajan, and A. Goldsmith, "The effect of time synchronization errors on the performance of cooperative miso systems," in *IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004.* IEEE, 2004, pp. 102–107.

[2] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, 2010.

[3] M. Goodarzi, N. Maletic, J. Gutiérrez, V. Sark, and E. Grass, "Next-cell prediction based on cell sequence history and intra-cell trajectory," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN).* IEEE, 2019, pp. 257–263.

[4] J. Zheng and Y.-C. Wu, "Joint time synchronization and localization of an unknown node in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1309–1320, 2009.

[5] M. Lévesque and D. Tipper, "A survey of clock synchronization over packet-switched networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2926–2947, 2016.

[6] G. Giorgi and C. Narduzzi, "Performance analysis of kalman-filter-based clock synchronization in ieee 1588 networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 8, pp. 2902–2909, 2011.

[7] M. Leng and Y.-C. Wu, "Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4860–4870, 2011.

[8] B. Lv, Y. Huang, T. Li, X. Dai, M. He, W. Zhang, and Y. Yang, "Simulation and performance analysis of the ieee1588 ptp with kalman filtering in multi-hop wireless sensor networks," *Journal of networks*, vol. 9, no. 12, p. 3445, 2014.

[9] M. Leng and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5404–5414, 2011.

[10] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, "Clock synchronization in wireless sensor networks: An overview," *Sensors*, vol. 9, no. 1, pp. 56–85, 2009.

[11] J. Eidson and K. Lee, "Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE.* Ieee, 2002, pp. 98–105.

[12] G. Gaderer, S. Rinaldi, and N. Kero, "Master failures in the precision time protocol," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2008, pp. 59–64.

[13] S. Ruffini, P. Iovanna, M. Forsman, and T. Thyni, "A novel sdn-based architecture to provide synchronization as a service in 5g scenarios," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 210–216, 2017.

[14] H. Li, L. Han, R. Duan, and G. M. Garner, "Analysis of the synchronization requirements of 5g and corresponding solutions," *IEEE Communications Standards Magazine*, vol. 1, no. 1, pp. 52–58, 2017.

[15] B. Etzlinger, H. Wymeersch, and A. Springer, "Cooperative synchronization in wireless networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pp. 2837–2849, 2014.

[16] A. Pásztor and D. Veitch, "Pc based precision timing without gps," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1. ACM, 2002, pp. 1–10.

[17] A. L. Barker, D. E. Brown, and W. N. Martin, "Bayesian estimation and the kalman filter," *Computers & Mathematics with Applications*, vol. 30, no. 10, pp. 55–77, 1995.

[18] F. R. Kschischang, B. J. Frey, H.-A. Loeliger *et al.*, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[19] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[20] M. Forbes, "Factor graphs and loopy belief propagation implemented in python," https://github.com/mbforbes/py-factorgraph/blob/master/README.md, 2017.