PTP-LP: Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP

Henning Puttnies, Peter Danielis, Dirk Timmermann
University of Rostock, Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Email: henning.puttnies@uni-rostock.de

***Abstract*—Clock synchronization protocols such as the precision time protocol (PTP), which are used to synchronize components of distributed systems, are fundamental to enable timed and coordinated activities, e.g., in real-time applications within the industrial Internet of things (IIoT). In theory, PTP is able to achieve a precision on the order of nanoseconds. However, its practical accuracy remains limited by packet delay variations. In this paper, we hence present a novel approach to increase the synchronization precision of PTP. Our approach (PTP-LP) relies on PTP to obtain precise hardware timestamps taken during multiple synchronization periods. These timestamps establish the constraints for a Linear Programming (LP) solver that is used to estimate the clock differences between devices. Moreover, we propose the heuristic PTP-H that achieves comparable accuracy but is less computationally complex. We evaluate PTP-LP and PTP-H in comparison with two state-of-the-art approaches under various conditions in terms of clock stabilities and packet delay distributions. PTP-LP and PTP-H are fully compatible with existing standards and show to be in particular robust to varying packet delays. Especially, PTP-LP outperforms previous approaches in presence of a stable hardware clock and unknown non-negligible network delay, which are both realistic working conditions.**

I. Introduction

Clock synchronization is used for providing a common time base to components within a distributed system. This is essential for timed and coordinated activities in real-time critical industrial Internet of things (IIoT) applications [1], [2] such as networked control loops in a smart factory or distributed sensing (where the sensed data need precise timestamps for further processing).

Hence, clock synchronization is a vital research field and there is a tremendous interest in the standardization of clock synchronization protocols. The most common clock synchronization protocols are the network time protocol (NTP) [3], the precision time protocol (PTP) [4], and the generalized precision time protocol (gPTP) [5] as part of the latest IEEE time-sensitive networking (TSN) standards [6] that aim at supporting real-time critical IIoT applications.

In general, synchronization protocols send packets to exchange timestamps between a master device and a slave device in order to estimate the path delays between both. Afterwards, they either adapt the slave clock to the (reference) master clock or remember the time difference between master and slave without correcting the slave clock, e.g., if the slave clock needs to be monotonic (especially, backward jumps can be problematic in some applications [7]). The synchronization precision of an approach is calculated as the remaining time difference between master clock and slave clock that cannot be compensated by the synchronization. As PTP by default assumes a constant and symmetric delay, its practical accuracy suffers if there is a non-negligible delay in the network. This delay might occur due to the fact that two devices synchronizing by means of PTP are connected via standard network infrastructure (e.g., switches that do not support prioritization or MAC-layer synchronization). Although this scenario is fully compliant to the PTP standard, it would lead to an insufficient synchronization precision. Especially, this might be important for distributed sensing, like in the W7-X fusion plasma experiment [8]), where standard network infrastructure shall be used for cost-efficiency and synchronization accuracy is still important.

Therefore, we propose using LP to mitigate the influence of network delay variations. Subsequently, we describe PTP-LP in a nutshell:

- First, we use PTP to send synchronization packets and include timestamps into them.
- Second, to increase the robustness towards delay variations, we formulate the slave clock function determination problem (SCFD-P) to estimate the slave clock function $C(t)$ referring to the master clock function $T(t)$. We use the timestamps taken during multiple synchronization periods as constraints for the SCFD-P.
- To solve the SCFD-P, we formulate two distinct linear subproblems by means of LP: one to determine the upper and one to determine the lower bound of $C(t)$. The PTP timestamps serve as constraints for these subproblems.

Clock synchronization has two main challenges: to compensate the inaccuracies of the slave clock and the communication channel that both are not ideal in reality. Sources of errors concerning the slave clock are quantization and frequency changes like jitter (random variation at each tick), wander (random walk of frequency), temperature effects (at greater timescales), and aging effects (at even longer timescales). Sources of errors concerning the communication channel are variable network delays (queuing) and propagation delays. We consider PTP-LP particularly significant concerning this matter as it exploits implicit constraints on timestamp values to improve clock parameter estimation. The contributions of this paper can be summarized as follows:

- PTP-LP is fully compatible to PTP, since we do not change the interface towards the PTP master. However, we utilize the timestamps of PTP on the slave devices all of which execute PTP-LP to achieve highest accuracy.

- We apply an additional estimation step to PTP. Even though applying a further estimation by means of a Kalman filter was already proposed in [9], to the best of our knowledge there is no existing approach that applies LP. However, using LP has a conceptual advantage: as the networking delay is composed of a fixed wire delay and a variant queuing delay, we can use the synchronization packets having the lowest delay as boundaries for the estimation of the slave clock. Therefore, LP is much more robust to delay variations than averaging or filtering. In addition, we propose the heuristic PTP-H that is more computationally efficient and mostly comparable to PTP-LP but always more precise than PTP.
- We conduct an evaluation considering different clock stabilities as well as different packet delay distributions. For this means, we use practical real-world measurements of self-similar delay distributions and clock stabilities.
- We compare PTP-LP and PTP-H with two state-of-the-art approaches: standard PTP and PTP with additional Kalman filtering from [9]. In comparison, PTP-LP shows to be much more robust against delay variations.

II. Related work

In this section, we present several clock synchronization approaches that are either common standards or promising research works. Many research approaches are not compatible with standard protocols. In all of the considered approaches in this section, a varying packet delay reduces the synchronization precision tremendously. In contrast, PTP-LP is compatible to existing standards (PTP) and shows to be robust against packet delay variations.

The Network Time Protocol (NTP) [3] is the most widely used synchronization protocol in the Internet. As shown in [10], the synchronization precision of NTP decreases if non-negligible delay variations occur. Furthermore, the accuracy of the NTP software timestamps is lower than the accuracy of hardware timestamps (e.g., PTP timestamps), which leads to a further decrease of the synchronization precision.

PTP [4] is the most common synchronization protocol that uses hardware timestamps. Consequently, the IEEE TSN group used PTP to derive gPTP as TSN substandard [5]. PTP-LP bases on PTP as it utilizes PTP timestamps. Although PTP can achieve a high synchronization precision, is assumes the packet delay between master and slave to be constant and symmetric. Therefore, packet variations can significantly reduce the synchronization accuracy. As shown in the evaluation, PTP-LP can handle these inaccuracies by applying an additional estimation step using LP. Note that an additional processing step (e.g., exponential averaging) would increase the robustness towards delay variations. However, averaging is not as robust as LP that exploits implicit constraints.

The IEEE TSN substandard gPTP [5] demands all switches to support gPTP synchronization in their MAC layers. In contrast, PTP-LP does not need special switching hardware, as it allows to compensate the delay variations on the network nodes. Therefore, PTP-LP might be also an interesting extension to the gPTP standard.

In [9], the authors present a much-noticed approach combining PTP and Kalman filtering that mitigates the errors that are generated by several uncertainties. However, the authors do not evaluate the number of synchronization periods used for the synchronization and the Kalman filter estimation (these evaluations would be important for the convergence and stability of the Kalman filter). Furthermore, the Gaussian variations have to be known a priori for the adjustment of the Kalman filter to ensure its precision and stability. Furthermore, they only consider a constant packet delay, which is far from a realistic scenario. We show that PTP-LP outperforms the approach from [9] especially if there is a high network load.

In [11], the authors propose an approach that is statistically robust and suited for passive (one-way communication) clock synchronization in wireless sensor networks (WSNs). Messages are timestamped at a sending measuring node, transmitted to a central reference node, and timestamped again. The authors propose a timing estimation by using the heavy tailed likelihood function to sort out outliers. As outliers basically falsify the Kalman filter estimation, the authors achieve a higher precision compared to a Kalman approach. In contrast to their approach, we not only sort out outliers but the LP solver inherently finds the most reliable timestamps.

The authors in [12] propose PulseSync that is suitable for large-scale networks. PulseSync floods the network with rapid and short pulses. This results into a short initialization phase and quick adaption of the synchronization to topological changes and clock drift. The authors state that they outperform the de facto standard for WSNs, the Flooding Time Synchronization Protocol (FTSP). Although the authors emphasize the generalizability of their approach, it highly relies on the WSN MAC layer (e.g., regarding the timestamps). Moreover, PulseSync is suboptimal regarding robustness as it exhibits a single point of failure (the reference node). In contrast, PTP-LP inherits from PTP the best master clock (BMC) algorithm, which ensures the compensation of a failing reference node.

Mallada et al. propose an approach without explicit estimation of the skew (frequency offset) in [7] and demonstrate its supremacy over NTP. The timestamps base on the Timestamp Counter (TSC) that counts CPU cycles since last restart and time measurements use an improved ping pong mechanism. The proposed algorithm uses the current offset information as well as an exponential average of the past offsets. This avoids the keeping of a long offset history as well as expensive computations. Besides these benefits, an exponential average is prone to delay variations in contrast to our LP-based approach that uses multiple timestamps to estimate the offset.

III. Clock model

In this section, we briefly describe the clock model, which we use in this paper and especially in the evaluation section. This discrete time clock model was proposed in [9] and bases on the continuous time clock model from [13]. It is a complex, non-linear clock model that considers uncertainties of the time offset increment and the frequency offset.

First, we consider the master time and slave time to follow linear functions $T(t)$ and $C(t)$, respectively (see Eq. 1). We consider $T(t)$ as reference clock and $C(t)$ to be a linear function of t with slope γ. Γ denotes the frequency offset between master and slave (see Eq. 2). Furthermore, we define the time offset θ as difference between $C(t)$ and $T(t)$.

$$T(t) = t, C(t) = \gamma \cdot t + \theta(0) \tag{1}$$

$$\gamma = \frac{f_{slave}}{f_{master}} \Rightarrow \Gamma = \gamma - 1 \tag{2}$$

As clocks follow non-linear functions in reality, we will consider this in the following. In particular, we assume a perfect master clock as it serves as reference and all non-linearities of the master clock are modelled as part of the slave clock $C(t)$ that is a stepwise linear function with stochastic noise. We assume the time offset θ and the frequency offset Γ to have random Gaussian uncertainties. Eq. 3 and 4 show w_θ and w_Γ, which denote the discrete time offset fluctuation and frequency fluctuation, respectively. Thereby $N(m, sd)$ refers to a Gaussian distribution with the mean m and the standard deviation sd. Moreover, σ_θ (and σ_Γ) denote the standard deviations of the offset (and frequency) fluctuation. Furthermore, s denotes the current clock step (e.g., clock tick) of $C(t)$ and δT refers to the time between two steps.

$$w_\Gamma(s) = N\left(0, \sqrt{\sigma_\Gamma^2 \cdot \delta T}\right) \tag{3}$$

$$w_\theta(s) = N\left(0, \sqrt{\sigma_\theta^2 \cdot \delta T}\right) \tag{4}$$

By taking into account these uncertainties, we can derive our clock models for the time offset θ (Eq. 5) and frequency offset Γ (Eq. 6) as functions of s assuming that Γ has an impact on time offset θ (as θ changes over time depending on Γ).

$$\theta(s+1) = \theta(s) + \Gamma(s) \cdot \delta T + w_\theta(s) \tag{5}$$

$$\Gamma(s+1) = \Gamma(s) + w_\Gamma(s) \tag{6}$$

IV. IEEE 1588: Precision Time Protocol (PTP)

PTP estimates both the time offset and the frequency offset between a master clock $T(t)$ and a slave clock $C(t)$ using several timestamps. Fig. 1 depicts both clocks as well as all PTP timestamps for one synchronization period. All variables will be introduced in the following. As the accuracy of the timestamps is crucial, precise hardware timestamps (e.g., MAC-layer) are preferred over software timestamps (e.g., taken at the application layer). Consequently, many devices support taking PTP hardware timestamps at their MAC layers.

A synchronization packet is created at the master and the sending moment is timestamped as T_1. $C(T_2)$ is the moment when the slave receives this packet. After a waiting time, the slave sends a synchronization packet back to the master and timestamps the sending moment as $C(T_3)$. T_4 is the moment when the master receives this packet. PTP assumes the path delay between master and slave on the forward d_{ms} and the reverse path d_{sm}, respectively, to be symmetric ($d_{ms} = d_{sm} = d$). Using the timestamps T_1 and T_4 (taken at the master) as well as $C(T_2)$ and $C(T_3)$ (taken at the slave), PTP estimates the time offset $\hat{\theta}_{PTP}(n)$ as given in Eq. 7. Thereby, n denotes the index of the synchronization period.

$$\hat{\theta}_{PTP}(n) = \frac{[C(T_2) - T_1] - [T_4 - C(T_3)]}{2} \tag{7}$$

Furthermore, PTP estimates the frequency offset $\hat{\Gamma}_{PTP}(n)$, as apparent from Eq. 8, from the differences of the time offsets in two consecutive synchronization periods. Thereby, T_1^n denotes the timestamp T_1 taken in the n-th synchronization period.

$$\hat{\Gamma}_{PTP}(n) = \frac{\hat{\theta}_{PTP}(n) - \hat{\theta}_{PTP}(n-1)}{T_1^n - T_1^{n-1}} \tag{8}$$

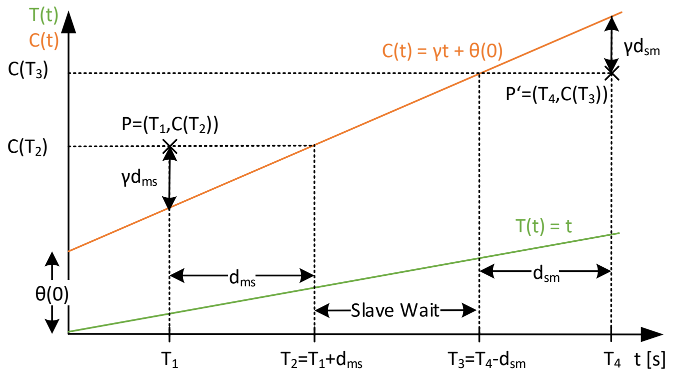

Fig. 1. Times at master $T(t)$ and slave $C(t)$ with the PTP timestamps for one synchronization period

V. PTP-LP: Using linear programming to improve the synchronization precision of PTP

PTP-LP relies on PTP to obtain precise hardware timestamps and applies an additional estimation to these. We estimate the linear function that the slave clock $C(t)$ follows using the master clock $T(t)$ as reference. More precisely, we estimate the slope γ and the y-intercept $\theta(0)$ of $C(t)$. We use the timestamps T_i^n taken at the master and $C(T_i^n)$ taken at the slave in several synchronization periods n and formulate an LP to find two linear functions: one above and one below the original slave clock function $C(t)$. We estimate $C(t)$ as the mean of both linear functions. As apparent from Fig. 1, the point $P = (T_1, C(T_2))$ must always be located above $C(t)$ and the point $P' = (T_4, C(T_3))$ must always be below $C(t)$. PTP-LP uses this knowledge to estimate $C(t)$ more precisely.

A. *Problem formulation*

To find the slope γ and the y-intercept $\theta(0)$ of $C(t)$, we define the clock function determination problem (SCFD-P) that uses the timestamps T_1^n, $C(T_2^n)$, $C(T_3^n)$, and T_4^n of several synchronization periods n as constraints. Here, T_1^n denotes the timestamp T_1 taken during the n-th synchronization period.

1) Forward path: find linear upper bound for C(t): We use the PTP timestamps T_1^n and $C(T_2^n)$ taken on the forward path of the synchronization packets from the master to the slave to find $f_{ub}(t)$ as linear upper bound for $C(t)$.

$$\alpha_1 = \gamma \tag{9}$$
$$\beta_1 = \theta(0) + \gamma \cdot d_{ms} \tag{10}$$

As apparent from Fig. 1, the slave clock line $C(t)$ is always below the point $P = (T_1, C(T_2))$. Consequently, $C(t)$ must be located below the point (T_1^n, C_2^n) for an arbitrary synchronization period n. We search for the slope α_1 and y-intercept β_1 of the linear upper bound $f_{ub}(t)$ as given in Eq. 9 and 10. As we state that the points $(T_1^n, C(T_2^n))$ are part of a (non-linear) upper bound for $C(t)$, we formulate the LP subproblem for the forward path using the constraints given in Eq. 11 to find $f_{ub}(t)$ as linear upper bound for $C(t)$.

$$\alpha_1 T_1^n + \beta_1 \leq C(T_2^n), \forall n \in [1, ..., N] \tag{11}$$

This basically means that every point of $f_{ub}(t)$ must be located below or on the set of points $(T_1^n, C(T_2^n))$ that serve as constraints since we know that $C(t)$ must run below $(T_1^n, C(T_2^n))$. Moreover, we can state the minimization function for this LP as given in Eq. 12.

$$f(\alpha_1, \beta_1) = \sum_{n=1}^{N} (C(T_2^n) - \alpha_1 T_1^n - \beta_1) \tag{12}$$

This basically means that $f_{ub}(t)$ should be as close as possible to the set of constraint points $(T_1^n, C(T_2^n))$. Thus, $f_{ub}(t)$ converges towards the constraint points.

2) Reverse path: find linear lower bound for C(t): Although the reverse path equations are very close to the forward path equation, we will state them here for the sake of comprehensibility. We use the PTP timestamps T_4 and $C(T_3)$ taken on the reverse path from slave to master to find $f_{lb}(t)$ as the linear lower bound for $C(t)$. We can state its slope α_2 and y-intercept β_2 as given in Eq. 13 and 14.

$$\alpha_2 = \gamma \tag{13}$$
$$\beta_2 = \theta(0) - \gamma \cdot d_{sm} \tag{14}$$

As apparent form Fig. 1, $C(t)$ is always located above the point $P' = (T_4, C(T_3))$. Consequently, $C(t)$ must be located above the point $(T_4^n, C(T_3^n))$ for an arbitrary synchronization period n. Since we state that the points $(T_4^n, C(T_3^n))$ are part of the (non-linear) lower bound for $f_{lb}(t)$, we formulate the LP problem as given in Eq. 15 for the reverse path by using the constraints to find $f_{ul}(t)$ as linear lower bound for $C(t)$.

$$\alpha_2 T_4^n + \beta_2 \geq C(T_3^n), \forall n \in [1, ..., N] \tag{15}$$

This basically means that every point of $f_{lb}(t)$ must be located above or on the set of constraint points $(T_4^n, C(T_3^n))$ as we know that $C(t)$ must run above $(T_4^n, C(T_3^n))$. Moreover, we can state the minimization function in Eq. 16.

$$f(\alpha_2, \beta_2) = \sum_{n=1}^{N} (\alpha_2 T_4^n + \beta_2 - C(T_3^n)) \tag{16}$$

This basically means that $f_{lb}(t)$ should be as close as possible to the set of constraint points $(T_4^n, C(T_3^n))$. Thus, $f_{lb}(t)$ converges towards the constraint points.

3) Determination of C(t) from lower and upper bound: Finally, the original $C(t)$ is estimated as mean of the upper and lower bounds f_{ub} and f_{lb}, respectively (Eq. 17).

$$\hat{\gamma}_{LP} = \frac{\alpha_1 + \alpha_2}{2}, \hat{\theta}_{LP}(0) = \frac{\beta_1 + \beta_2}{2} \tag{17}$$

4) PTP-H: a heuristic for PTP-LP: We also propose a heuristic denoted as PTP-H to find $\hat{f}_{ub}(t)$ and $\hat{f}_{lb}(t)$ as approximations for $f_{ub}(t)$ and $f_{lb}(t)$ in a computationally more efficient way. Let $P_1 = \{((T_1^1, C(T_2^1)), ..., (T_1^n, C(T_2^n))\}$ be the set of forward path constraints points and $f_{lr1}(t)$ the linear regression for all points in P_1. Then $\hat{f}_{ub}(t)$ equals to:

$$\hat{f}_{ub}(t) = f_{lr1}(t) - m_1 \tag{18}$$
$$m_1 = \max_{(T_1^i, C(T_2^i)) \in P_1} (f_{lr1}(T_1^i) - C(T_2^i)). \tag{19}$$

Respectively, let $P_2 = \{(T_4^1, C(T_3^1)), ..., (T_4^n, C(T_3^n))\}$ be the set of reverse path constraints points and $f_{lr2}(t)$ the linear regression for all points in P_2. Then $\hat{f}_{lb}(t)$ equals to:

$$\hat{f}_{lb}(t) = f_{lr2}(t) + m_2 \tag{20}$$
$$m_2 = \max_{(T_4^i, C(T_3^i)) \in P_2} (C(T_3^i) - f_{lr2}(T_4^i)). \tag{21}$$

We basically shift the regression line towards the constraints point that is closest to $C(t)$ (see Fig. 1). PTP-H completely substitutes the LP solver.

5) Conceptual errors of PTP-LP: Observe that we use a complex, realistic, and non-linear clock model. However, our LP-based approach assumes a linear function for $C(t)$. Due to this mismatch, a systematic error is introduced into the model. Nevertheless, this error can be neglected if $C(t)$ can be assumed to be linear for a sufficiently short period of time. As show in [14] the clock changes more quickly when high temperature gradients are applied. As a consequence, the LP solver might not find a solution within a short period of time and the solving is stopped due to a timeout. Furthermore, many synchronization packets lead to a long synchronization time. After a long period of time, the non-linearity of the slave clock function $C(t)$ becomes relevant. This might lead to the situation that there is no solution as the original clock is non-linear and there might be no linear clock function that fulfills all constraints for $C(t)$. However, in this case it is still suitable to choose a lower synchronization period and therefore improve the accuracy of PTP-LP.

VI. Evaluation

In this section, we discuss the evaluation results obtained through numerical simulations using Python 3.6.2 and SciPy 1.0.0 on a desktop PC (i7-3770, 32GB RAM). We compare PTP-LP and PTP-H to standard PTP [4] and PTP extended by Kalman filtering [9]. We choose PTP for comparison in this evaluation as it is the most widely used synchronization protocol that utilizes precise hardware timestamps and hence serves as a benchmark. We further conduct a comparison to

TABLE I
CLOCK PARAMETERS USED IN THE EVALUATION.

	σ_θ^2	σ_γ^2	$\sigma_{C(t)}^2$
HW clock	1e-14	1e-18	1e-18
SW clock	1e-12	1e-16	1e-6

the Kalman filtering approach [9], as it is the most promising and well-known synchronization approach that extends PTP.

In general, the synchronization accuracy depends on clock stability [9] and network delay. Therefore, the accuracy depends on the devices and traffic patterns in the distinct application scenario. Hence, we cover a variety of different stabilities and delay distributions in this evaluation.

A. Methodology

The synchronization period was set to one second (default for PTP [4]). We executed every experiment $M = 100$ times.

We compare the different approaches using two metrics: the synchronization error and the frequency error. The synchronization error is the remaining difference between master and slave time that cannot be compensated by the synchronization approach. The frequency error $F_{err} = |F_{est}/F_{ref}|$ is the absolute value of the quotient of the estimated slave clock frequency F_{est} and of the actual slave clock frequency F_{ref}.

For the evaluation, we use the clock model from Sec. III. To model different clock stabilities, we use several clock parameters: variance of the time offset noise at the slave device σ_θ^2, variance of the frequency offset noise at the slave device σ_γ^2, and variance of the timestamp noise at the slave device $\sigma_{C(t)}^2$. We use clock stabilities of two classes: a HW clock (e.g., a hardware counter that is clocked by a hardware oscillator) and a SW clock (e.g., a software counter that is incremented using software interrupts). Table I depicts the values of the clock parameters used in the evaluation. They correspond to real-world clocks measured [15] and [16]. Moreover, we depict the Allan variance $\sigma_y^2(\tau)$ of both clocks in Fig. 2 ($\sigma_y^2(\tau) = \frac{\sigma_\theta^2}{\tau} + \frac{\sigma_\gamma^2 \cdot \tau}{3}$, cf. [9]). For the variance of the timestamp noise at the slave device $\sigma_{C(t)}^2$, we had to make own assumptions. We assume an accuracy on the order of nanoseconds for the standard deviation of the timestamps obtained from the HW clock. For the timestamps obtained from the SW clock, we assume an accuracy of the standard deviation on the order of milliseconds.

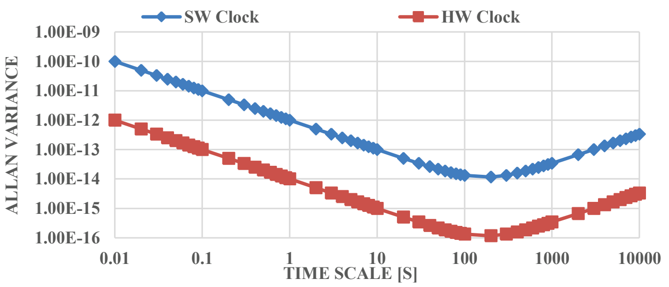

Fig. 2. Allan variances for SW and HW clock used in the evaluation.

B. Packet delay distributions used in the simulations: Gaussian delay and self-similar delay

In this section, we describe the examined packet delay distributions (Gaussian and self-similar).

The Gaussian delay distribution assumes a simple Gaussian delay probability with a mean delay of 5ms and a standard deviation of 2ms. However, the self-similar network traffic is more realistic according to [17]. Thereby, the term self-similar means that the probability distribution looks similar for different time scales [17]–[19]. The Gaussian delay distribution is easy to calculate (using the mean delay and standard deviation).

In contrast, we conduct a complex calculation of the self-similar delay in our simulations. Thereby, we assume the packet delay to be the sum of the propagation and the queuing delay. We further assume the propagation delay to be constant and the queuing delay to be a non-uniformly distributed statistical process. We use the self-similar packet arrival time distributions from [17] to calculate a realistic queuing delay. We scale the packet arrival distributions to a certain link utilization and we calculate the queue fill level $fl_{queue}(t)$ at a switch port using a packet size $size_{packet}$ of 800 bytes, which was the mean packet size observed in [17]. Therefore, we assume it as a realistic value for generic background traffic (note that the packet size has only a minor impact on the simulation results). The queuing delay d_{packet} of a synchronization packet is calculated from $fl_{queue}(t)$ at the arrival time $t_{arrival}$ of the synchronization packet and the transmission speed s_{trans} (e.g., 1 GBit/s for GBit Ethernet): $d_{packet} = fl_{queue}(t_{arrival})/s_{trans}$. Actually, we calculate the time that is needed to empty the queue after $t_{arrival}$ as this time equals the queuing delay of this synchronization packet.

C. Evaluation with Gaussian packet delay

As we have a non-negligible network delay in this evaluation, PTP does not achieve an accuracy on the order of nanoseconds. PTP-LP achieves a higher accuracy than PTP under all examined conditions in this section (see Fig. 3).

There are no major differences between the HW and SW clock as the delay uncertainties are more dominant than the clock uncertainties in this scenario. Actually, for PTP-LP and PTP-H the outliers of the Gaussian delay often result into one bound (upper or lower) to be much closer to the $C(t)$ than the other bound. This results into errors as we estimate $C(t)$ as mean of both bounds.

When examining the SW clock, the Kalman filter achieves a higher accuracy than PTP-LP regarding the estimation of the frequency error (see Fig. 3). The reasons for this are twofold. First, this results from the Gaussian uncertainties (noise) of time offset, frequency offset, and timestamps taken at the slave device using a SW clock. These Gaussian uncertainties can be compensated by the Kalman filter, which is by design optimal for Gaussian noise. Moreover, the Kalman filter can also compensate the packet delay in this experiment as it follows a Gaussian distribution. However, note that the Gaussian variations have to be known a priori for the adjustment of the Kalman filter to ensure its precision and stability. Second, the uncertainty of the SW clock at the slave device results into a non-linear slave clock function $C(t)$, which cannot be handled by PTP-LP and PTP-H.

When examining the HW clock, we can conclude that PTP-LP outperforms the Kalman-based approach (see Fig. 3). The reason for this is that the very low uncertainty of the HW clock results into a sufficiently linear slave clock function $C(t)$. Thus, LP is very suitable to estimate it.

The accuracies of PTP-LP and PTP-H increase if more packets are used for the synchronization. The reason behind that is that more PTP synchronization packets provide more timestamps and hence more information for the LP solver. As a consequence, it is easier for the LP solver to estimate time offset and frequency offset as the impact of the uncertainties due to the delay variations is reduced. PTP-H achieves a precision that is even slightly better than PTP-LP as the used LP solver does not always find the optimal solution for the LP and the linear regression also benefits from more information.

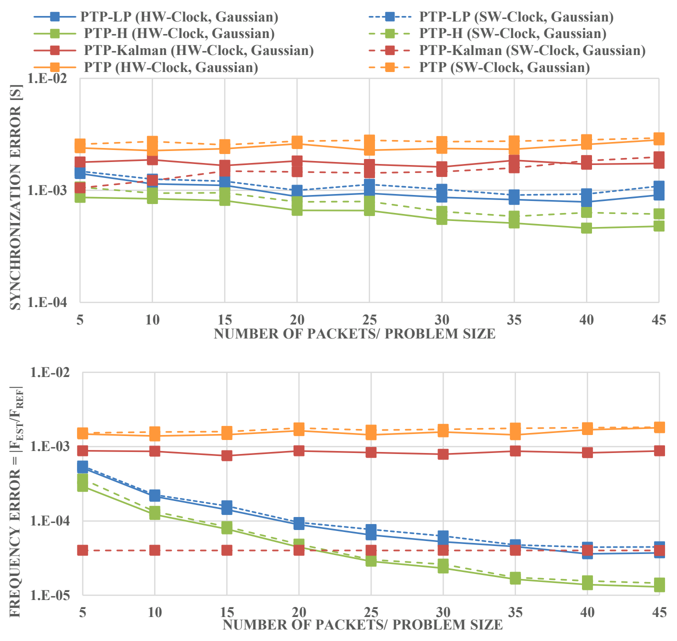

Fig. 3. Synchronization error and frequency error for Gaussian network delay (the graph is logarithmic in the Y axis).

D. *Evaluation with self-similar network delay*

Furthermore, we conducted an evaluation using a self-similar delay distribution for the synchronization packets.

We chose fixed values of approx. 10% and 90% for the utilization as an evaluation of various values would enormously expand the size of the evaluation. Moreover, the actual utilization heavily dependents on the scenario, which complicates general statements. However, by using a high network utilization, we can show that PTP-LP still achieves a high synchronization precision under these difficult conditions. The HW clock leads to a higher accuracy than the SW clock as the clock uncertainties are more dominant than the delay uncertainties in this scenario. Moreover, the HW clock is sufficiently linear and thus PTP-LP and PTP-H are very suitable to estimate it.

Fig. 4 depicts the results for 10% link utilization. There are no major differences, but PTP-LP and PTP-H perform slightly better than PTP and PTP-Kalman.

In contrast, the self-similar delay distribution with 90% link utilization leads to large errors that cannot be compensated by standard PTP or the Kalman filter. PTP-LP achieves a higher accuracy (see Fig. 5) as it can handle delay uncertainties that do not follow a Gaussian distribution since the LP solver is able to compensate any kind of a delay distribution. PTP-H performs not as good as PTP-LP as PTP-H evaluates only the largest distance to a constraint point, which is usually one at the beginning or the end. In contrast, PTP-LP evaluates all points equally to find an optimal solution. Again, the accuracy of PTP-LP and PTP-H increases if more packets are used for the synchronization, as they can use more information and estimate more precisely. Moreover, the influence of statistical uncertainties decreases.

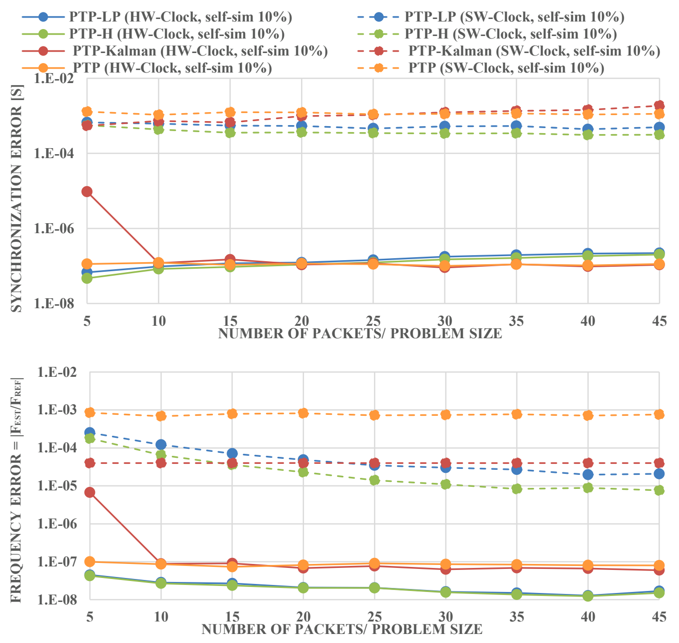

Fig. 4. Synchronization error and frequency error for self-similar network delay with 10% link utilization (the graph is logarithmic in the Y axis).

E. *Computational complexity*

Fig. 6 depicts the computation time of PTP-LP and PTP-H for different numbers of synchronization packets (corresponding to the problem size to be solved). LP is known to have a polynomial complexity. Accordingly, the computation time seems to be polynomial depending on the problem size. PTP-H has a much lower computation time than PTP-LP. However, the solving time of the LP was always below 10ms. Although the LP solving is an additional estimation step that introduces a computational overhead, we can increase the synchronization accuracy. Consequently, PTP-LP offers to trade-off between

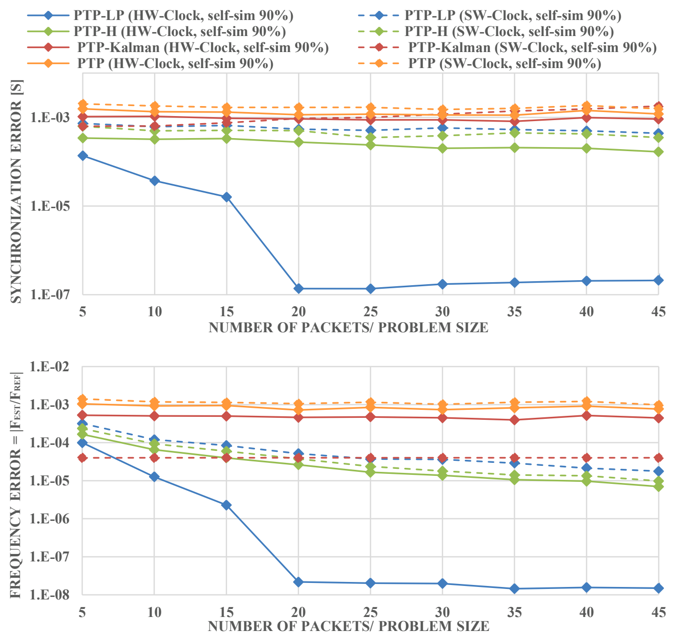

Fig. 5. Synchronization error and frequency error for self-similar network delay with 90% link utilization (the graph is logarithmic in the Y axis).

computation time and accuracy. Nevertheless, PTP-H also seems to offer a good trade-off between computation time and precision and might be suitable for constrained devices.

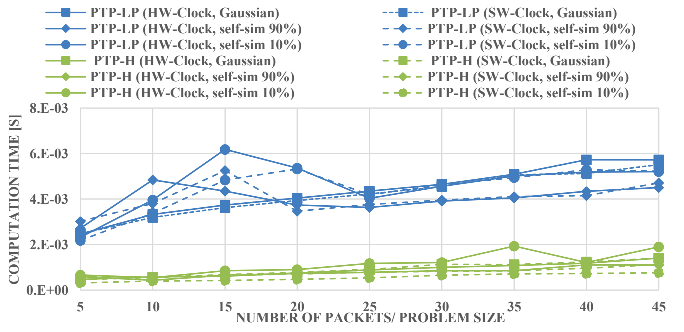

Fig. 6. Computation time for PTP-LP and PTP-H (the graph is logarithmic in the Y axis).

VII. Summary and Conclusion

In this paper, we presented an LP-based approach to increase the precision of PTP that is particularly robust against packet delay variations. PTP-LP is fully compatible to the common standard PTP since we do not change the interface towards the PTP master. However, we utilize the PTP timestamps on the slave device. Furthermore, we propose the heuristic PTP-H that achieves comparable accuracy but is less computationally expensive.

We conducted a comprehensive evaluation considering different clock stabilities as well as different packet delay distributions. In comparison with two state-of-the-art approaches (standard PTP [4] and PTP with Kalman filtering [9]), the proposed approaches PTP-LP and PTP-H outperform both. Especially, PTP-LP shows to be robust against delay variations in the case of a high link utilization.

References

[1] T. Qiu, Y. Zhang, D. Qiao, X. Zhang, M. L. Wymore, and A. K. Sangaiah, "A Robust Time Synchronization Scheme for Industrial Internet of Things," *IEEE Trans. on Industrial Informatics*, p. 1, 2017.

[2] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[3] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification."

[4] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[5] ——, *IEEE standard for local and metropolitan area networks*, ser. International standard. Geneva and s.l. and New York: ISO, 2014, vol. ISO/IEC/IEEE 8802-1AS:2014(E). [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=6739979

[6] ——, "Time-Sensitive Networking Task Group," 24.01.2018. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[7] E. Mallada, X. Meng, M. Hack, L. Zhang, and A. Tang, "Skewless Network Clock Synchronization Without Discontinuity: Convergence and Performance," *IEEE/ACM Trans. on Networking*, vol. 23, no. 5, pp. 1619–1633, 2015.

[8] R. C. Wolf, A. Ali, A. Alonso, J. Baldzuhn, C. Beidler, M. Beurskens, C. Biedermann, H.-S. Bosch, S. Bozhenkov, R. Brakel *et al.*, "Major results from the first plasma campaign of the Wendelstein 7-X stellarator," *Nuclear Fusion*, vol. 57, no. 10, p. 102020, 2017.

[9] G. Giorgi and C. Narduzzi, "Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks," *IEEE Trans. on Instrumentation and Measurement*, vol. 60, no. 8, pp. 2902–2909, 2011.

[10] A. Bletsas, "Evaluation of Kalman filtering for network time keeping," *IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 52, no. 9, pp. 1452–1460, 2005.

[11] J. O. Nilsson and P. Händel, "Robust recursive network clock synchronization," in *IEEE ICONECCT*, 2014, pp. 1–5.

[12] C. Lenzen, P. Sommer, and R. Wattenhofer, "PulseSync: An Efficient and Scalable Clock Synchronization Protocol," *IEEE/ACM Trans. on Networking*, vol. 23, no. 3, pp. 717–727, 2015.

[13] C. Zucca and P. Tavella, "The clock model and its relationship with the Allan and related variances," *IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 52, no. 2, pp. 289–296, 2005.

[14] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *IPSN '10 Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks 2010*, 2010, p. 151.

[15] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the IEEE 1588 precision time protocol," in *Conference on IEEE*, vol. 1588, 2005, pp. 11–15.

[16] H. Abubakari and S. Sastry, "IEEE 1588 style synchronization over wireless link," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2008, pp. 127–130.

[17] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879175

[18] B. B. Mandelbrot, *The fractal geometry of nature*. New York, NY: Freeman, 1983.

[19] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. on Networking*, vol. 2, no. 1, pp. 1–15, 1994.