Clock Synchronization Using Linear Programming, Multicasts, and Temperature Compensation

Henning Puttnies, Eike Schweissguth, Dirk Timmermann
University of Rostock, Institute of Applied Microelectronics and CE
Rostock, Germany, Email: henning.puttnies@uni-rostock.de

Jörg Schacht
Max Planck Institute of Plasma Physics
Greifswald, Germany

***Abstract*—Clock (or time) synchronization is essential for many applications in the Industrial Internet of Things (IIoT). Hence, it is a vital research field and important field of standardization ambitions. The most accurate protocols like PTP and gPTP need specialized hardware to reach their maximum precision. Without this hardware, they cannot compensate massive packet delays. It was shown that approaches based on linear programming (LP) can mitigate this problem. However, changes in the clock frequency lead to nonlinear clocks, which are not well compensated by LP-based approaches. As a consequence, we propose the SLMT approach that uses LP, multicasts, and temperature compensation for time synchronization. To the best of our knowledge, SLMT is the first synchronization approach that combines LP and one-way exchange or multicasts, respectively. Consequently, it is efficient regarding the number of messages. Furthermore, to the best of our knowledge, SLMT is the first synchronization approach that combines LP with a temperature compensation in order to mitigate LP's conceptual drawback with nonlinear clocks. In an extensive evaluation and comparison to many state-of-the-art approaches, we show that SLMT outperforms these approaches, especially under harsh conditions like rapid temperature changes and unknown non-negligible network delays.**

I. Introduction

The term clock (or time) synchronization refers to providing a common time base to devices, e.g., in a distributed system. This is an essential prerequisite for coordinated activities like networked control loops or distributed sensing, that are typical for Industrial Internet of Things (IIoT) scenarios. Actually, there is a vast number of clock synchronization applications [1]. Hence, time synchronization is a vital research field and there is a tremendous interest in the standardization of synchronization protocols. Very common protocols are the Network Time Protocol (NTP), Precision Time Protocol (PTP), and generalized Precision Time Protocol (gPTP) as part of the IEEE's latest Time-Sensitive Networking (TSN) standards.

Generally, an approach synchronizes the time of a slave device with a master (or reference) time. The two main challenges in the clock synchronization domain are how to compensate the inaccuracies of the slave clock and of the communication channel, as both are not ideal in reality. Sources of inaccuracies and nonlinearities of the slave clock are: quantization, frequency changes (e.g., due to temperature effects [1], [2]), random variations at each tick (jitter), random walk of frequency (wander), and aging effects at long timescales. Sources of inaccuracies and variations in the communication channel are: variable network delays (e.g., due to queuing) and variable processing delays in the software stacks. The precision of a synchronization approach is the remaining time difference that cannot be compensated by the synchronization.

In this paper, we propose a novel approach called time synchronization using linear programming, multicasts, and temperature compensation (SLMT). Compared to approaches like PTP and gPTP, SLMT can achieve a very high precision without using special switching hardware. SLMT does not need hardware timestamps. However, it could use them, if they are available. Moreover, the gPTP standard states that gPTP's precision is restricted to 7 hops. In contrast, our LP-based approach is very suitable for large topologies with unknown infrastructure (e.g., virtual private networks (VPNs) over the Internet). Note that SLMT is a novel and holistic approach. However, the idea of combining LP with temperature compensation is also a valid extension of PTP (or PTP-LP [3], respectively). [1]

Possible applications of SLMT are, on the one hand, large distributed measurement sites like the W7-X fusion experiment [4]. Here, measurements should be timestamped as precisely as possible. Moreover, the standard switching hardware should be as simple as possible in order to reduce costs and improve sustainability. On the other hand, VPNs over the Internet are a possible application for SLMT, if different measurement sites should be synchronized as precisely as possible.

In the following, we will describe the approach in a nutshell. SLMT is composed of two phases. In the *delay* phase, LP and two-way exchange of messages are used to estimate the minimum delay between master and slave. The *delay* phase has to be executed only once, if the network does not change. In the *sync* phase, LP and multicast-based one-way exchange of messages are used to synchronize master and slave. This leads to a high efficiency in terms of the number of messages. Therefore, SLMT is suitable even for very large networks. LP exploits implicit constraints on the timestamps to improve the clock parameter estimation. Thus, LP-based synchronization is very robust against delay variations as shown in [3], [5]. Furthermore, SLMT uses a temperature compensation to further improve the precision of the LP. Note that it is beneficial to decouple temperature compensation and LP, as temperature and clock nonlinearities are highly correlated [2], [6] and

[1] A prototype implementation could use PTP's message format, PTP's hardware timestamps, and a PTP software implementation (e.g., LinuxPTP or PTPd) as starting point.

temperature information are available on almost any device [6].

The contributions of the paper are as follows:

- To the best of our knowledge, we propose the first combination of LP and multicasts (or one-way exchange) in a time synchronization approach.
- Moreover, to the best of our knowledge, we propose the first combination of LP and temperature compensation for synchronization. This is important to mitigate a conceptual drawback of LP: it cannot compensate for nonlinearities of the slave clock, as LP estimates linear functions. As a consequence, it is sufficient to provide only one very stable temperature compensated oscillator (that typically costs more than 100$) for the master and use algorithmic temperature compensation for all slave devices.
- Finally, we conduct an extensive evaluation using different clock stabilities, network delay conditions, and several state-of-the-art approaches for comparison.

II. Related work

In this paper, we focus on wired scenarios as safety is crucial in the real-time IIoT, but it is still an open research question for wireless technologies [7].

NTP is a synchronization protocol that is widely used in the Internet. However, it is not suitable for real-time IIoT applications, as it is software-only and its precision decreases, if delay variations occur [5].

PTP commonly uses hardware timestamps to achieve a high precision. However, it assumes the network delay to be constant and symmetric. Therefore, delay variations reduce its accuracy, as measured, e.g., in [8]. Such variations can occur if switches do not have MAC layer support for PTP. In our evaluation, we show that LP-based approaches can handle these delay variations. It is possible to add additional processing to the timestamps (e.g., exponential filtering) to increase PTP's delay robustness. Nevertheless, we show that LP is even more robust. From PTP, the IEEE Time-Sensitive Networking (TSN) group derived gPTP as TSN substandard. In contrast to PTP, gPTP requires all switches to support gPTP at the MAC layer. However, our LP-based approach can compensate the network delay variations at the slave devices. Therefore, we do not need specialized switching hardware.

In [9], the authors propose an approach combining PTP and Kalman filtering (PTP-Kalman), in order to mitigate clock and delay uncertainties. One drawback of PTP-Kalman is that the uncertainties have to be known a priori for the adjustment of the Kalman filter to ensure its precision and stability. Moreover, Kalman filters are only optimal for Gaussian uncertainties. These are major problems in realistic scenarios, as the delay follows a self-similar behaviour [10] and clock nonlinearities correlate with temperature [11].

PTP-LP [3] is an approach combining PTP and LP. In contrast to PTP-LP, SLMT supports multicasts for synchronization that are more efficient. Moreover, SLMT's temperature compensation further improves the precision of the LP.

In [12], Yang et al. propose an approach based on an interactive multi-model Kalman filter (IMM) to estimate the clock skew (Yang10). An IMM consists of multiple Kalman filters. Each filter uses a different system model to estimate the skew. The IMM can determine the filter that fits to the skew measurements adaptively. As an extension, they propose EACS in [2] that additionally uses temperature information to estimate the skew. Here, one filter uses a constant temperature model ($T = const$) and the other one uses a constant temperature change model ($\delta T/\delta t = const$). Similarly, two Kalman filters estimate the skew, one uses a constant skew model and the other one a constant skew change model. The IMM chooses the model that most probably fits to the temperature. This information is used to choose the corresponding filter to estimate the skew. This approach works, as temperature and skew are highly correlated according to [2], [6]. [2] Actually, we combine Yang10 and EACS to an IMM-based temperature compensation and use it in one version of SLMT. As shown in our evaluation, SLMT outperforms Yang10 and EACS.

TCTS [6], [13] is computationally efficient and simple. It measures (either offline or online) the skew for a certain temperature and uses these measurements for skew compensation. In a second version of SLMT, we apply TCTS for temperature compensation and compare it with the IMM version.

HUYGENS [1] is an approach to achieve precise synchronization without specialized hardware. It filters noisy data and processes it using Support Vector Machines (SVM). Thus, a major difference is that HUYGENS uses SVMs and SLMT uses LP to estimate skews and offsets. However, in contrast to SLMT, HUYGENS does not apply a temperature compensation and assumes the clock to be piecewise linear. This leads to errors, if high temperature gradients occur as in [6].

The authors of [14] propose an approach for positioning in 5G networks that provides synchronization as a by-product. They propose a combined estimation of time of arrival and direction of arrival based on a cascaded structure of extended Kalman filters (EKF). The calculations take place in the network infrastructure. In contrast, SLMT's estimations are conducted on the slave devices and this decentralized approach is more scalable. Furthermore, the EKF would struggle with non-Gaussian delays in packet switched networks.

III. Notations

In this section, we define necessary terms like the master (or reference) clock $T(t)$ as a linear function as apparent from Eq. 1. Note that $T(t)$ is a perfect linear function with the slope of 1 as we model all nonlinearities as part of slave clock $C(t)$. $C(t)$ can also be modelled as a linear function having the slope γ and the y-intercept $\theta(0)$ (Eq. 1), if its frequency is stable. The time offset $\theta(t)$ is defined as difference between $T(t)$ and $C(t)$ (Eq. 2). Here, γ equals to the quotient of the clock frequency of the slave f_{slave} and the clock frequency of the master f_{master}. Moreover, Γ is the frequency difference

[2] [2] shows that using constant and constant change models is sufficient.

between both, normalized by the master clock frequency, and is referred to as frequency offset or skew (Eq. 3). We can measure the skew using two offset measurements with a time difference of ΔT.

$$T(t) = t, C(t) = \gamma \cdot t + \theta(0) \quad (1)$$

$$\theta(t) = C(t) - T(t) \quad (2)$$

$$\gamma = \frac{f_{slave}}{f_{master}} \Rightarrow \Gamma = \gamma - 1 = \frac{\theta(t + \Delta T) - \theta(t)}{\Delta T} \quad (3)$$

IV. The SLMT Approach

In [2], [6], [12], it was shown that it is possible to compensate temperature and skew. In [3], [5], it was shown that LP can compensate the delay. The main idea of SLMT is to combine both in one approach.

SLMT's message flow has two phases (cf. Fig. 1). As the network delay consists of a constant part (e.g., propagation delay) and a probabilistic part (e.g., queuing delay), we can estimate the constant part once (*delay* phase) and use efficient multicasts for the actual synchronization (*sync* phase). During the *delay* phase, the slave sends an initial message to the master (forward path). The sending moment in time is timestamped as $C(T_1)$ and the receiving moment in time is timestamped as T_2. On the reverse path from the master to the slave, the sending moment in time is timestamped as T_3 and the receiving moment in time is timestamped as $C(T_4)$. Note that $C(T_1)$ and $C(T_4)$ are taken with respect to the slave clock and the actual times at the master T_1 and T_4 are unknown. SLMT uses the knowledge that $P = (T_2, C(T_1))$ must always be below $C(t)$ and $P' = (T_3, C(T_4))$ must always be above $C(t)$ to estimate $C(t)$. As depicted in Fig. 1, P or P' are very close to $C(t)$ if the delay is very small. Consequently, if a packet encounters the minimum delay, the LP uses this information to increase its precision. During the *sync* phase, the master sends multicast messages to all slaves (sync path). The sending moment in time is timestamped as T_5 and the receiving moment in time is timestamped as $C(T_6)$. Here, we use the knowledge that $P'' = (T_5, C(T_6))$ must be above $C(t)$.

Furthermore, SLMT applies a temperature compensation to the slave clock. The benefit is as follows. LP can compensate the probabilistic delay completely, if one point $((T_i, C(T_i))$ is as close as $y \cdot d_{min}$ to $C(t)$. Where d_{min} is the minimum constant part of the packet delay. However, LP's precision decreases if the slave clock is nonlinear. Therefore, SLMT linearizes the slave clock using a temperature compensation in order to improve the results of the LP. The temperature compensation is executed in parallel during the *delay* and *sync* phase. If the slave cannot control its clock frequency, it estimates a temperature compensated virtual clock $C'(t)$ and the timestamps at the slave are taken with respect to $C'(t)$. [3]

A. Problem Formulation and LP

SLMT can be described as follows. It tries to estimate $C(t)$ using $T(t)$ as reference. Actually, it estimates $C(t)$'s

[3]The memory overhead is very low, as the slave only saves the offset between its physical clock $C(t)$ and its virtual clock $C'(t)$.

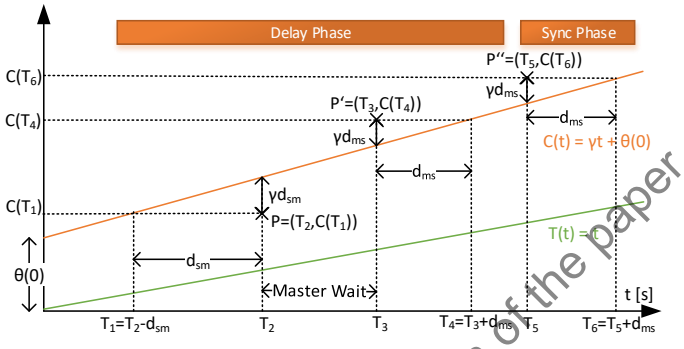

Fig. 1. Times at master $T(t)$ and slave $C(t)$ with the timestamps (Note: for the sake of simplicity this Fig. shows the timestamps of only one period)

γ and $\theta(0)$. Therefore, we formulate the clock function determination problem (CFD-P). For this, we use timestamps $C(T_1^n)$, T_2^n, T_3^n, $C(T_4^n)$, T_5^n, $C(T_6^n)$, taken over N periods with $n \in [1, ..., N]$, as constraints.

1) Delay Phase - Forward Path and Lower Bound, Reverse Path and Upper Bound: As the first step of the *delay* phase is similar to [3], [5], we skip a detailed description of the estimation of $f_{lb}(t)$ and $f_{ub}(t)$, which are $C(t)$'s lower and upper bounds.

2) Delay Phase - Estimation of C(t) and Propagation Delay: Similar to [5], we estimate $C(t)$ as $\hat{C}_{delay}(t)$ using $f_{lb}(t)$ and $f_{ub}(t)$ (cf. Eq. 4). The approach from [5] ends here. In contrast, we use $\hat{C}_{delay}(t)$ to estimate the minimum delay between master and slave. More precisely, we estimate the product $\gamma \cdot dmin_{ms}$ that is its projection to the y-axis (cf. Eq. 5 and 6).

$$\hat{\gamma}_{delay} = \frac{\alpha_1 + \alpha_2}{2}, \hat{\theta}_{delay}(0) = \frac{\beta_1 + \beta_2}{2} \quad (4)$$

$$P_{rev} = \{(T_3^1, C(T_4^1)), ..., (T_3^N, C(T_4^N))\} \quad (5)$$

$$\gamma \cdot dmin_{ms} = \min_{(T_3^n, C(T_4^n)) \in P_{rev}} (C(T_4^n) - \hat{C}_{delay}(T_3^n)) \quad (6)$$

By this, we use the set of constraint points from the reverse path P_{rev} and calculate the minimal difference between a point in P_{rev} and $\hat{C}_{delay}(t)$ (cf. Fig. 1).

3) Sync Phase - Upper Bound: In the *sync* phase, we can use multicasts and a one-way message exchange to estimate $C(t)$ again. Actually, we estimate a new upper bound for $C(t)$ and move it down by $\gamma \cdot dmin_{ms}$. The result of the LP is $\hat{C}_{sync}(t) = \hat{\gamma}_{sync} \cdot t + \hat{\theta}_{sync}(0)$, an estimation of $C(t)$ (cf. Eq. 9). Therefore, the LP minimizes Eq. 8 under the constraints of Eq. 7. This basically means that $\hat{C}_{sync}(t) + \gamma \cdot dmin_{ms}$ should be as close as possible to the constraint points $(T_5^n, C(T_6^n))$.

$$\alpha T_5^n + \beta \leq C(T_6^n) - \gamma \cdot dmin_{ms}, \forall n \in [1, ..., N] \quad (7)$$

$$f(\alpha, \beta) = \sum_{n=1}^{N} (C(T_6^n) - \gamma \cdot dmin_{ms}) - (\alpha T_5^n + \beta) \quad (8)$$

$$\hat{\gamma}_{sync} = \alpha, \hat{\theta}_{sync}(0) = \beta \quad (9)$$

B. Temperature Compensation

For the temperature compensation, we apply two approaches. A complex approach using an IMM and the simple

TCTS approach [13]. Basically, both are existing approaches for temperature compensation. However, our novel idea is to use them to calculate $C'(t)$, a linearized version of $C(t)$. The slave permanently compensates the temperature and takes timestamps referring to $C'(t)$. However, these are processed as described in the previous sections.

1) IMM-Mixed: We propose a combination of Yang10 [12] and EACS [2] for an IMM-based temperature compensation called IMM-Mixed. IMM-Mixed uses the temperature IMM and structure of EACS. However, it uses the skew models of Yang10, as they consider the time offset and skew. Moreover, we propose using the measurement noise matrices from [9] as they are more accurate. IMM-Mixed alone is a novel synchronization approach. However, we do not evaluate it individually without the LP due to limited space.

2) TSTC as Heuristic: The simple TCTS approach proposed by [**?**], [13] is to measure (either offline or online) the skew for a certain temperature and use this measurement value to compensate $C(t)$'s skew.

V. Clock Models

In this section, we describe the realistic clock models used for our evaluation. As we conducted numerical simulations, the slave clock would be ideal by default. In order to achieve realistic results, we implemented clock models that reproduce the clock inaccuracies of real devices.

A. Random Walk Clock Model

Giorgi et al. proposed the random walk clock model in [9]. Here, the offset of the next clock tick $\theta(s+1)$ is calculated using the offset of this clock tick $\theta(s)$, the skew $\Gamma(s)$, and the clock step width T_{tick}. Additionally, w_θ is a Gaussian uncertainty at every offset iteration (e.g., a jitter of the clock counter's increment period). Similarly, the skew $\Gamma(s+1)$ is calculated using the skew of this clock tick $\Gamma(s)$ and w_Γ, which is a Gaussian uncertainty (e.g., a slow change in the frequency that accumulates over time):

$$\theta(s+1) = \theta(s) + \Gamma(s) \cdot T_{tick} + w_\theta(s) \tag{10}$$

$$\Gamma(s+1) = \Gamma(s) + w_\Gamma(s). \tag{11}$$

B. Temperature Clock Model

We propose a new temperature clock model. Referring to [2], the temperature is the most relevant reason for skew changes. The offset calculation is similar to the random walk clock model. However, we model the skew $\Gamma(T)$ as a function of the temperature T and add a Gaussian uncertainty w_Γ.

$$\Gamma(s+1) = \Gamma(T) + w_\Gamma(s) \tag{12}$$

C. Temperature or Skew Compensation

As for the compensation of temperature (or skew, respectively) introduced in Sec. IV-B, we can either correct the skew or the offset to calculate $C'(t)$. [4] For offset correction, Eq. 10 changes as follows:

$$\theta(s+1) = \theta(s) + (\Gamma(s) - \hat{\Gamma}(T)) \cdot T_{tick} + w_\theta(s). \tag{13}$$

[4]Simultaneously correcting skew and offset would be duplicative.

TABLE I
Overview of all approaches examined in the evaluation.

Abbreviation	Description
SLMT	Uses LP and multicasts (used with random walk clock model)
SLMT-NTC	SLMT without temperature compensation
SLMT-IMM	SLMT using an IMM for temperature compensation
SLMT-TH	SLMT using TCTS as heuristic for temperature compensation
PTP	Precision Time Protocol
PTP-EF	PTP with additional exponential filtering
PTP-Kalman [9]	PTP with additional Kalman filtering
PTP-LP [3]	PTP with additional linear programming
Yang10 [12]	IMM for clock synchronization
EACS [2]	IMM for clock synchronization using temperature information

Here, $\Gamma(T)$ is the skew as a function of the temperature T and $\hat{\Gamma}(T)$ its estimation (e.g., given by TSTC or IMM-Mixed). For skew correction, Eq. 12 changes as follows:

$$\Gamma(s+1) = \Gamma(T) - \hat{\Gamma}(T) + w_\Gamma(s). \tag{14}$$

VI. Evaluation

Table I depicts an overview of all approaches examined in the evaluation.

A. Methodology

We assume that the master and the slave are connected via a switch and there is also an additional device that generates background traffic. This device models the entire background traffic in the network and can generate traffic even higher than the wire speed. We used numerical simulation, conducted every experiment 100 times, and calculated the mean synchronization error (offset error) and the mean frequency error (skew error). As synchronization period, we used one second (i.e., default for PTP). As clock step width T_{tick}, we used one millisecond. We varied the number of periods (or packets, respectively) used for the *delay* and *sync* phase.

1) Clock stabilities: We used the clock models described in Sec. V. For the random walk clock model, we used the same values for w_θ and w_Γ as in [3], [9]. As shown in [3], this results into Allan Variances measured on real devices.

For the temperature clock model, we used the temperature skew correlation measured in [15]. For the experiments, we change the temperature from 20°C to 50°C. Such temperature changes where measured in [6], especially in case of fast sunlight shade transitions in outdoor scenarios. As we evaluate various numbers of periods, the simulated time t_{sim} varies and we scale the temperature track. The starting point of the transition is at $0.3 \cdot t_{sim}$ and the terminal point at $0.7 \cdot t_{sim}$. As a consequence, a higher t_{sim} leads to a slower temperature change.

2) Delay Probabilities: Self-similar delay is realistic according to [10]. This means that the probability distribution looks similar on different time scales [10]. The calculation is complex but similar to [3]. The delay of a packet $delay_{packet}$ equals to $delay_{prop} + delay_q$. Here, $delay_{prop}$ is constant.

However, $delay_q$ is modelled as the time that is needed to empty the queue after the arrival time of a packet $t_{arrival}$: $delay_q = fl_q(t_{arrival})/s_{trans}$. To calculate queue fill level $fl_q(t)$ at a switch port, we used the self-similar packet arrival time distribution from [10] and scaled it to a certain mean link utilization. For this, we used a packet size of 800 Byte (mean in [10]) and a transmission speed s_{trans} of 1 GBit/s (for GBit Ethernet):

B. Evaluation Using the Random Walk Clock Model

Due to the limited space, we only show diagrams for a relatively stable clock (HW clock, e.g., a hardware counter). When using a less stable clock, the trends are similar and the LP-based approaches are still better than the state-of-the-art (SOTA) approaches. Nevertheless, the results of SLMT and PTP-LP are not as good as for the HW clock, as the LP struggles with nonlinear clocks.

For 10% mean link utilization, there are no major differences between the approaches regarding the frequency error (cf. Fig. 2). Regarding the synchronization error, the approaches PTP-Kalman, PTP-LP, and SLMT perform much better than PTP and PTP-EF (PTP with exponential filtering).

For 90% mean link utilization, the diagrams show that PTP-LP and SLMT can achieve major improvements (cf. Fig. 2). The reason is that LP is very robust to delay variations. PTP-LP performs slightly better than SLMT, as it uses two-way unicasts and SLMT uses one-way multicasts for synchronization. Consequently, PTP-LP can use more information to estimate more precisely. However, SLMT's one-way multicasts result in much less overhead. There is one outlier for SLMT at 30 synchronization packets, as there is the rare case that $C(t)$ is too nonlinear to be estimated by the LP. However, this is not a real drawback of SLMT, as it can simply use less packets for the LP in this case. Especially, since the precision is stable for more than 20 packets.

C. Evaluation Using the Temperature Clock Model

As LP struggles with nonlinear clocks, we propose a temperature compensation and evaluate it using the temperature clock model (cf. Fig. 3).

For 10% mean link utilization, PTP-LP performs comparable to SLMT-WTC (SLMT without temperature compensation). The reason is, that both are comparable LP-based approaches without temperature compensation. However, SLMT-IMM (using the IMM) and SLMT-TH (using the temperature heuristic TCTS) perform much better, as they compensate the temperature by linearizing the clock. SLMT-IMM and SLMT-TH have a comparable performance, but SLMT-TH performs even slightly better. The reasons are the different skew measurement schemes and the corresponding response times of the approaches. SLMT-TH measures the temperature skew correlation offline. Note that we do not assume SLMT-TH's skew measurements to be perfect. Instead, we determined the measurement uncertainty experimentally using self-similar delay and 50% mean link utilization. However, SLMT-TH can compensate the skew directly, if the temperature changes.

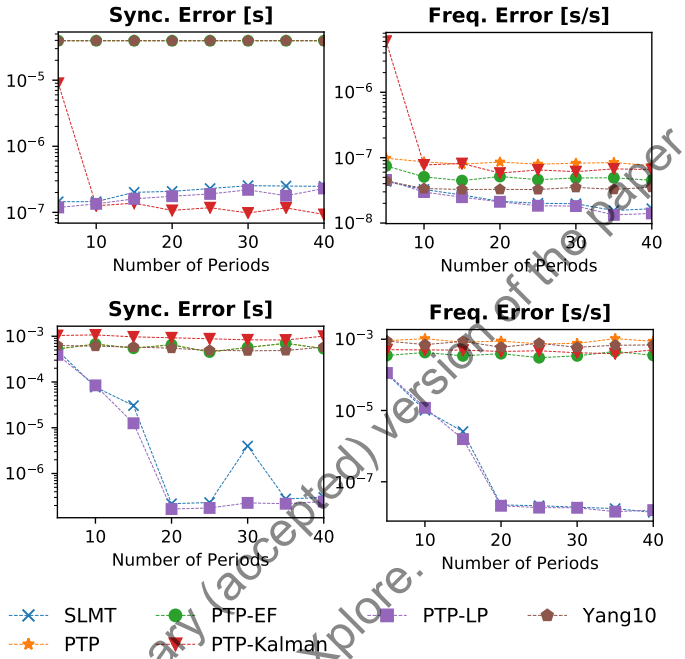

Fig. 2. Results for random walk clock. The upper graphs are for 10% mean link utilization and the lower for 90%.

In contrast, the IMM measures the skew online. As one skew measurement needs two offset measurements (cf. Eq. 3), every online approach needs one synchronization period to measure the skew and compensate it. To mitigate this, one can reduce the synchronization period. [5] Considering the frequency error, SLMT-IMM and SLMT-TH are less precise than other approaches. The reason is, that the actual skew is zero due to the temperature compensation. However, the LP tries to estimate the actually still slightly nonlinear slave clock, which leads to these errors. The more packets are used, the more $C(t)$ follows approx. a linear function and the precision of the LP rises. In order to mitigate this, we can assume that the skew is always zero due to the temperature compensation and let SLMT-IMM or SLMT-TH compute the synchronization error.

For 90% mean link utilization, the diagram shows that SLMT-TH achieves a much smaller error than all other approaches. The reason is that SLMT-TH can still linearize the clock based on its offline measurements. In contrast, SLMT-IMM's IMM has serious problems as the skew estimations are very inaccurate due to the high delay variations. As a result, the slave clock function is distorted and not linearized.

D. Computational Complexity

Fig. 4 depicts the mean solving times for various approaches. Note that these solving times are very uncritical, as it takes, e.g., 45 seconds to collect 45 packets, which is multiple orders of magnitude larger than the solving time. Interestingly,

[5] The reason for the slightly increasing synchronization error after approx. 25 packets is as follows: a slower and longer temperature track leads to a stronger nonlinearity of the slave clock, which is harder to compensate for.

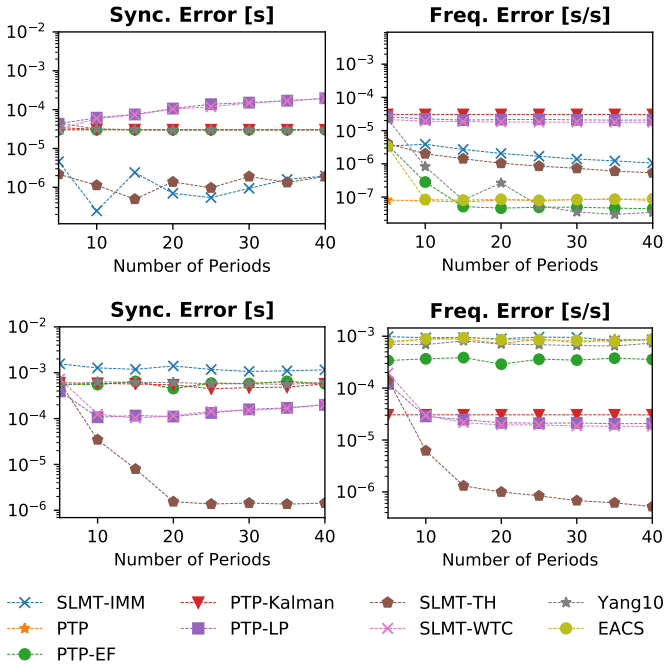

Fig. 3. Results for temperature clock. The upper graphs are for 10% mean link utilization and the lower for 90%.

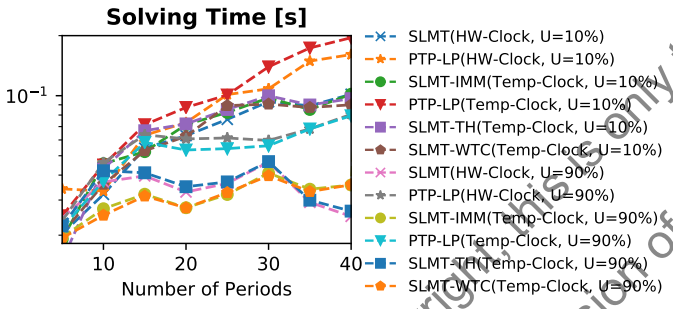

Fig. 4. Computation time for various approaches.

the temperature compensated approaches are faster than the ones without temperature compensation, as a linearized slave clock results into a simpler LP.

VII. Summary and Conclusion

In this paper, we propose the SLMT approach that uses linear programming, multicasts, and temperature compensation for clock synchronization. To the best of our knowledge, SLMT is the first synchronization approach that combines linear programming (LP) and one-way exchanges (or multicasts, respectively). Furthermore, to the best of our knowledge, SLMP is the first synchronization approach that combines LP and temperature compensation.

In an extensive evaluation and comparison to many SOTA approaches, we show that SLMT outperforms the SOTA, especially under harsh conditions, e.g., temperature changes and unknown non-negligible network delays. We can conclude the following novel findings. Firstly, the combination of multicasts and LP is possible and promising, as it reduces the overhead traffic for synchronization compared to the traditional two-way synchronization and only leads to minor reduction of the precision. Secondly, also the combination of LP and temperature compensation is possible and promising. We can conclude that the temperature heuristic SLMT-TH is simple and robust even under a high network load. In contrast, using a complex IMM for temperature compensation works well and without prior knowledge under low and medium network loads. However, its precision is degrading under harsh network conditions as Kalman filters cannot compensate the non-Gaussian delay uncertainties that lead to skew measurement errors.

SLMT is an algorithmic approach. An implementation is possible in hardware or software. Furthermore, a hybrid version might be interesting using PTP's hardware timestamps and computing the LP as well as the temperature compensation in software. Moreover, we consider the combination of temperature compensation and LP also as a valuable extension to PTP or PTP-LP [3], respectively.

References

[1] Y. Geng *et al.*, "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *USENIX NSDI 2018*.

[2] Z. Yang *et al.*, "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *IEEE INFOCOM 2012*.

[3] H. Puttnies *et al.*, "PTP-LP: Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP," in *IEEE GLOBECOM 2018*.

[4] R. C. Wolf *et al.*, "Major results from the first plasma campaign of the Wendelstein 7-X stellarator," *Nuclear Fusion*, vol. 57, no. 10, 2017.

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

[6] T. Schmid *et al.*, "High-resolution, low-power time synchronization an oxymoron no more," in *ACM/IEEE International Conference on Information Processing in Sensor Networks 2010*.

[7] A. Mahmood *et al.*, "Clock synchronization over IEEE 802.11—A survey of methodologies and protocols," *IEEE Trans. on Industrial Informatics*, vol. 13, no. 2, 2017.

[8] K. S. Lee *et al.*, "Globally synchronized time via datacenter networks," in *ACM SIGCOMM*. ACM, 2016.

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

[10] T. Benson *et al.*, "Network Traffic Characteristics of Data Centers in the Wild," in *ACM SIGCOMM 2010*.

[11] M. Jin *et al.*, "DualSync: Taming clock skew variation for synchronization in low-power wireless networks," in *IEEE INFOCOM 2016*.

[12] Z. Yang *et al.*, "Adaptive Clock Skew Estimation with Interactive Multi-Model Kalman Filters for Sensor Networks," in *IEEE ICC 2010*.

[13] T. Schmid *et al.*, "Temperature Compensated Time Synchronization," *IEEE Embedded Systems Letters*, vol. 1, no. 2, 2009.

[14] M. Koivisto *et al.*, "Joint device positioning and clock synchronization in 5G ultra-dense networks," *IEEE Trans. on Wireless Communications*, vol. 16, no. 5, 2017.

[15] Z. Yang *et al.*, "Temperature-Assisted Clock Synchronization and Self-Calibration for Sensor Networks," *IEEE Trans. on Wireless Communications*, vol. 13, no. 6, 2014.