

A Simulation Framework for Evaluating Wireless Sensor Clock Discipline Algorithms

André Lübken

Department of Avionics Systems

Institute of Space Systems, German Aerospace Center (DLR)

28359 Bremen, Germany

{andre.luebken}@dlr.de

Abstract—Wireless Sensor Networks (WSNs) deployed in harsh environments, such as space missions, face the challenge of maintaining precise time synchronization, especially due to dynamic conditions like rapid temperature fluctuations. This paper introduces a novel, tick-based simulation framework designed to model and evaluate the performance of local clock discipline algorithms. The framework’s methodology is based on a clock error model that incorporates inherent drift, periodic variations, and stochastic noise. An event system simulates dynamic environmental impacts, such as temperature changes derived from actual satellite mission data, allowing for realistic test conditions. In addition, an evolutionary optimization approach is implemented to tune algorithm parameters. To demonstrate its utility, a case study is presented comparing several algorithms using a scenario derived from satellite telemetry. The results validate the simulation framework as a flexible tool for algorithm development and parameter tuning, facilitating the prediction of algorithm performance under extreme conditions.

Index Terms—wireless sensor networks, wireless communications, simulation, time synchronization, clock discipline

I. INTRODUCTION

Wireless Sensor Networks (WSNs) have become integral to a wide range of critical domains, i.e., in industrial automation and aerospace systems. In these applications, distributed nodes collaboratively sense and act upon their environment. The ability to fuse data from multiple sensors, guarantee deterministic latency, and coordinate actions depends on a common notion of time across the network. However, maintaining this time synchronization is particularly challenging in harsh environments where sensors are exposed to extreme and rapidly changing conditions.

In applications such as spacecraft and launch vehicles, wireless sensors are foreseen to be used in wireless avionics systems to save on weight and reduce design complexity [1]–[3]. In this application, sensors are subjected to severe environmental stress. Extreme temperature gradients, like those experienced by a satellite moving between direct sunlight and Earth’s shadow within a single orbit, can cause the commonly used quartz crystal oscillators that govern local clocks to drift unpredictably at varying rates. Factors like electromagnetic interference (EMI) and radiation can introduce additional, non-

linear errors, causing local clocks to rapidly lose synchronization with a reference time.

The conditioning of the local clock therefore forms the basis of a time synchronization system, as each node needs to derive the current time from its local clock in between the synchronization steps that happen on a higher level.

Addressing this challenge requires robust local clock discipline algorithms capable of compensating for the drift effects. However, validating the performance of such algorithms presents a significant problem. Testing on physical hardware necessitates the use of expensive and time-consuming thermal vacuum chamber tests for space applications. On a higher layer, for time distribution protocols, existing network simulation platforms like NS-3 or OMNeT++ [4] are suitable for modeling packet-level network phenomena such as routing influence on time distribution. For the lower level local oscillator discipline, no tool exists to model the specific physics of clock drift in extreme environments and to evaluate the performance of local clock discipline algorithms.

This paper introduces a novel simulation framework designed specifically to evaluate local clock discipline algorithms. The core purpose of this framework is to provide a common, reproducible baseline for developing, optimizing, and calibrating algorithms under realistic, data-driven scenarios before committing to expensive hardware testing. The system is based on a formal clock error model and a dynamic event system capable of ingesting real-world mission data.

The remainder of this paper is structured as follows: Section II reviews related work in time synchronization and simulation. Section III details the clock behavior modeling, followed by a description of the simulation frameworks architecture in section IV. Section V presents a case study evaluating multiple algorithms using synthetic and real-world satellite data. Finally, Section VI concludes this paper.

II. RELATED WORK

The development of the simulation framework for local clock discipline is based on two areas of research: the evolution of adaptive synchronization algorithms designed to operate in dynamic environments and the capabilities of existing simulation tools used to validate them. Essential groundwork has been established [5] where more suitable approaches to time synchronization in WSNs have been proposed.

A. Adaptive Algorithms for Environmental Compensation

As WSNs are deployed in harsher conditions, research has shifted towards algorithms that can adapt to environmental disturbances, primarily temperature. In less demanding scenarios, a sufficiently advanced adaptive algorithm might mitigate the need for extensive pre-deployment simulation.

Initial approaches were purely reactive. The method proposed by Stanislawski et al. for IEEE 802.15.4e networks, e.g., triggers a packet-based re-synchronization whenever the local temperature changes by more than a set threshold [6]. Inherently, this method acts only after a thermal change has occurred, potentially allowing significant error to accumulate.

A more sophisticated, predictive approach involves learning a model of the clock's behavior. The work by Martinez et al. demonstrates this by constructing a "learning table" that maps observed drift rates to local temperature measurements [7]. This allows a node to estimate its current drift based on its temperature. However, this approach is limited as it fails to account for the state of the reference clock, which is also subject to thermal drift.

Another approach is proactive feed-forward control, as demonstrated by the Adaptive Local Clock Control (ALCC) algorithm that sparked the development of the simulation framework discussed in this paper. This method uses a pre-characterized physical model of the oscillator's quadratic response to temperature. By measuring the temperature in real-time, it calculates the expected drift and applies a correction to cancel the disturbance. While the algorithm provides good results for its application domain, it is dependent on the preceding oscillator characterization and parameter tuning [8]. Verifying its performance under extreme conditions necessitates a simulation environment capable of modeling these conditions.

B. Simulation Frameworks for Time Synchronization

Many studies leverage general-purpose network simulators like OMNeT++ or ns-3. While powerful for analyzing network-level protocols, such as time distribution protocols, these tools typically abstract away the physics of hardware component effects.

To address this, more specialized frameworks have been developed. For instance, the TS2 simulator, built on OMNeT++, provides more detailed models for the IEEE 1588 Precision Time Synchronization Protocol (PTP), including drifting oscillators [9]. Similarly, co-simulation frameworks like Symphony integrate emulated hardware clock models with the option to simulate drifting oscillators by providing altered time ticks to the simulation based on e.g., linear or random drift [10]. They, however, do not include an event driven system to create the rapid changes that occur in extreme environments.

III. CLOCK BEHAVIOR MODELING

The simulation framework focuses on modeling an oscillator's behavioral profile by combining several sources of error.

This is achieved through two components: a unified clock error model and a dynamic event system.

The literature (e.g., in [11] or [12]) treats time synchronization as a first-order dynamics problem of the local clock.

$$\tau_i(t) = \alpha_i t + o_i \quad (1)$$

with t denoting the reference time and τ as the local clock time that is influenced by skew α and phase offset o_i .

Fig. 1 illustrates this for two local clocks i, j with different drift factors.

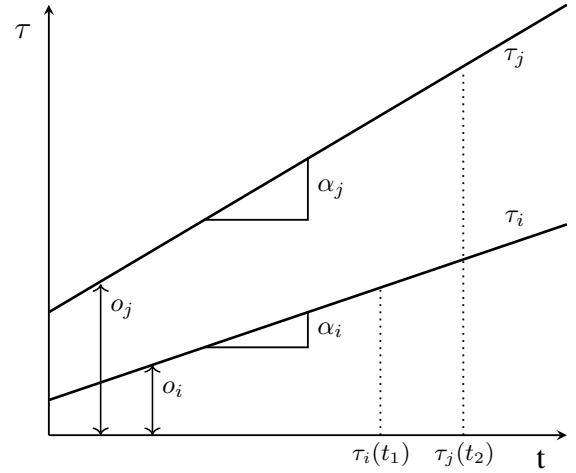


Fig. 1. Clock model of dynamics τ against absolute time t for local clocks i and j

A. Unified Clock Error Model

For each tick of the global simulation time, a local clock instance calculates its own notion of a time step, or local tick (T_l), which incorporates a configurable timing error. This value determines how much the local clock's time advances relative to the reference time. For every tick the simulation computes the local tick according to the following equation:

$$\tau = \frac{1 + \overbrace{(\sin(p) \cdot I)}^{\text{periodic drift}} + \overbrace{(X \cdot I \cdot d_{v1} \cdot d_{v2} \cdot d_f)}^{\text{random variance}} + e}{r} \quad (2)$$

where:

- 1 represents an ideal tick with no error.
- $\sin(p) \cdot I$ models a periodic drift component, where p is a factor incremented each tick and I is the inherent drift of the oscillator.
- $X \cdot d_{v1} \cdot d_{v2} \cdot d_f$ represents a stochastic noise component. Here, X is a random factor, d_{v1} and d_{v2} are variance factors for regular intervals, and d_f controls the overall influence of the random drift.
- e is the event-driven drift that represents the influence of external environmental factors. This value is injected by the dynamic event system..
- r is the result resolution, a scaling factor to control the granularity of the output data.

This model allows the framework to generate a baseline drift profile for each clock that incorporates a deterministic, periodic and random component. An example for 3 clocks simulated after this system is depicted in Fig 2.

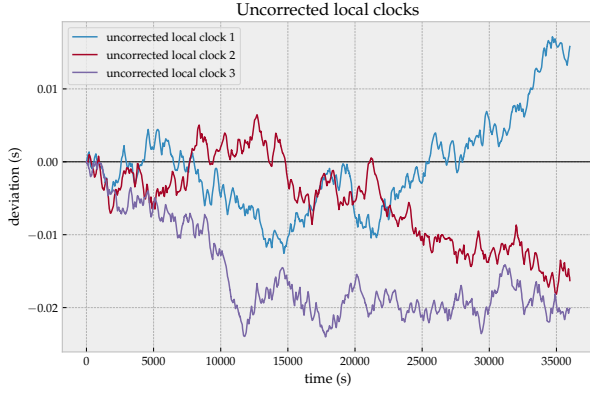


Fig. 2. Uncorrected drift of several simulated local clocks. The random variables used to generate the error components are based on a different random seed for each of them.

An Allen Deviation (ADEV) calculation was used to validate the model against a set of oscillator measurements that were done during the characterization phase. Compared to a root mean square error calculation, this approach is more suitable for oscillator characterization as it is less susceptible to errors introduced by outliers, deviations in constant drift components or phase alignment. The model validates a short term stability of a 1-second ADEV of $6.32 \cdot 10^{-6}$ and a flicker floor (best-case stability) of $8.24 \cdot 10^{-7}$ at $\tau = 100$ s. The reference mean of 6 characterized oscillators in a temperature controlled environment showed values of $6.87 \cdot 10^{-6}$ for the 1-second ADEV and $7.82 \cdot 10^{-7}$ for the flicker floor respectively.

This shows that the model performs well when validated against the behavior of a real world oscillator despite its relative simplicity compared to more elaborate approaches based on computationally expensive filters. This is done in order to allow the model to be executed on resource constrained devices like the ones deployed in a typical sensor network that might be operated under energy harvesting conditions, where energy-intensive floating point operations might reduce the usable sensing time of the network node.

B. Dynamic Event System

To simulate the impact of changing environmental conditions, a dynamic event system was implemented. It allows a simulation scenario to contain a time-ordered queue of events that are processed by the simulation controller at runtime. It implements the following events:

- **interval change:** This event modifies the synchronization interval, which is the period between when a local clock receives an update of the reference time. It is used to test an algorithm's adaptability and its performance during different periods without external correction.

- **drift:** This event introduces a fixed, user-defined drift value directly into the e term. It is used to simulate specific anomalies or to create sudden step-changes in the drift rate to test an algorithm's reaction time and stability.
- **temp:** An event to take a temperature value (e.g., from real-world data) as input. Using a pre-characterized polynomial function that describes a specific oscillator's frequency-versus-temperature curve, the event calculates the corresponding clock drift and injects it into the e term. A random variation is added to simulate the random noise inherent in a physical temperature sensor that would be used for temperature-based correction in a real world application.

IV. SIMULATION FRAMEWORK ARCHITECTURE

The architecture of the simulation framework is designed to be modular, to allow for easy extension of the core components of the system, i.e., to allow for the easy addition of new clock discipline algorithms for investigation. The runtime is highly configurable to support the optimization and reporting features of the framework.

A rudimentary implementation of this low level behavior is also possible in existing network simulators like OMNeT++. Nevertheless, they are optimized for packet level discrete event simulation and lack the flexibility and speed of a dedicated system. The system presented here in contrast allows the user to develop and optimize clock discipline algorithms in parallelizable parameter studies to create the foundation of time synchronization algorithms for harsh environments.

A. Core Components

The framework is built upon three primary components that work together. The simulation controller is the central component of the simulation. It is responsible for initializing the environment, managing the global simulation time (or "tick"), processing the time-ordered event queue, triggering the execution of each clock instance per tick, and logging all relevant data for post-simulation analysis.

The next component is the local clock instance. These are representations of individual sensor node local clocks. Each instance is an independent entity that maintains its own local time, drift characteristics, and a specific clock discipline algorithm under test (e.g., PID, ALCC, etc.). At each tick, the instance calculates its time advancement based on the parameters of the unified clock error model. Multiple clocks can be simulated within a single simulation scenario.

These components are tied together by the scenario configuration. To ensure reproducibility, each simulation run is defined by a human-readable configuration file in YAML format. This file specifies all global parameters, such as the total simulation duration and data logging resolution. It also defines the set of local clock instances to be simulated, the algorithm each one will use and the sequence of events to be processed by the Event System.

B. Execution Flow

A simulation run is processed in a tick-based loop:

- 1) **Initialization:** The simulation controller parses the scenario configuration file, instantiates the specified local clock instances, and populates the event queue from the scenario definition.
- 2) **Simulation Loop:** The controller iterates through the simulation ticks until the configured duration is reached. In each tick:
 - a) The global time is advanced.
 - b) The controller checks the event queue. If an event's timestamp matches the current global time, it is processed. For example, a `temp` event would cause the controller to update the `e` parameter for the target clock instance.
 - c) The controller iterates through each local clock instance, executing the advance of a time step. The instance calculates its new local time estimate based on its current error parameters and the discipline algorithm it implements.
 - d) If the current tick is a designated synchronization point, the controller provides the reference global time to the clock instance. The instance's algorithm can use this reference data point to compute a new correction.
 - e) At configured intervals, the controller logs the state of each clock instance, including its local time and its deviation from the ground truth.
- 3) **Termination:** Once the simulation duration is complete, the controller writes the data logs, which can then be used for analysis and visualization.

Since a simulation run can take a significant amount of time, depending on the number of clocks simulated and the scenario duration, results files are saved to disk by default. This way, the result data and graph generation can be performed separately to ease the development of the result analysis.

C. Evolutionary Optimization

A key goal of the framework is the integration of an automated parameter tuning system based on a genetic algorithm (GA). This allows the system to be used for the optimization of clock parameters for a given application scenario.

Genetic algorithms are a class of optimization techniques based on the concept of natural selection. They are particularly effective for problems with large solution spaces. They were chosen as an optimization method for the simulation framework due to their ability to handle disconnected parameters like the ones used in most time synchronization algorithms well and for their potential to avoid local optima. The algorithm is based on letting a population of candidate parameter sets evolve over many generations toward an optimal result. The evolution is driven by a fitness function, which evaluates how well each parameter set performs, and genetic operators like selection, crossover, and mutation, which create new pa-

rameter sets from existing ones to establish the next generation for testing. Fig. 3 illustrates this.

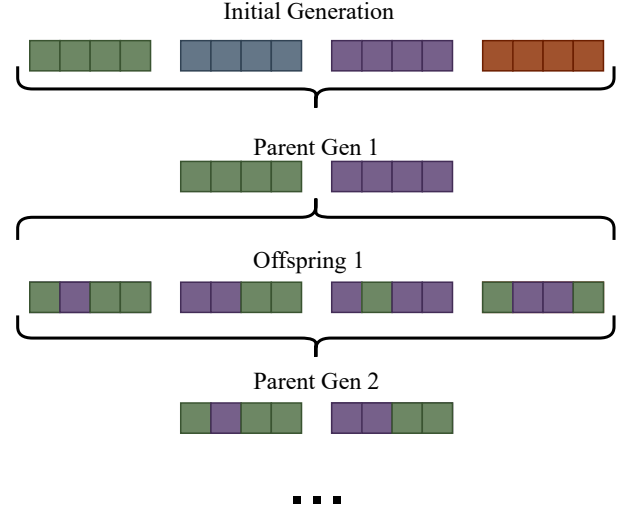


Fig. 3. General scheme of the genetic algorithm system used in the simulation framework

The GA is used to find the optimal set of tuning parameters for a given clock discipline algorithm (e.g., the K_p , K_i , and K_d coefficients for a simple PID controller). It is configured as another operational mode in the scenario configuration file. The process is as follows:

- **Chromosome:** A chromosome is the vector of the parameters to be optimized, for example, $[K_p, K_i, K_d]$.
- **Population:** The GA maintains a fixed, configurable set of these chromosomes, that are subsequently used to run the simulation scenario.
- **Fitness Function:** The fitness of a parameter set is determined by executing a simulation run using its parameter set. The fitness score in this scenario is the inverse of the mean and maximum deviation from the reference clock.
- **Evolutionary Cycle:** The parameter sets with the highest fitness value from the current population are selected as parent generations for the next cycle. New offspring (new parameter sets) are created by applying crossover (combining random parameters of the two parent parameter sets) and mutation (small, random changes to a parameter of a set). This new generation replaces the old, and the process repeats.

By running this evolutionary cycle for a set number of generations, the population converges toward an optimized set of parameters. For the example of the ALCC algorithm that is showcased here, good results were achieved with a population size of 500 with a cutoff after 200 generations and a mutation rate of 0.015. Computation time for this optimization took approximately 2 days on a desktop workstation with parallelized workload.

V. CASE STUDY: SYNTHETIC AND REAL WORLD DATA

To demonstrate the effectiveness of the simulation framework, a case study was conducted using synthetic and real-world scenarios. For a comparison of two algorithms, a setpoint weighted PID (SW-PID) and the ALCC algorithm were used. The temperature dependent drift was obtained from frequency deviation data of an oscillator characterized under thermal test chamber conditions.

A. Scenario Definition

The scenarios were designed to replicate thermally challenging environments for a sensor on a spacecraft. A subset of telemetry data from the Eu:CROPIS satellite mission was used for the real world scenario, specifically the temperature sensor readings from a solar array over several orbits. This data, which shows temperature swings from approximately -70°C to $+83^{\circ}\text{C}$, was converted into a stream of temperature events. The simulation was configured with an initial synchronization interval of 10 seconds, which was later changed to a much longer 600-second interval mid-simulation to test the algorithms' long-term stability. In addition, synthetic scenarios were created that use predefined changes in the drift rate of the simulated oscillators to create nascent drift conditions.

B. Results

Fig. 4 and 5 show the results for the two scenarios respectively. Both algorithms went through a GA optimization phase to find the optimal parameters for a synthetic standard scenario that incorporates rapid changes in drift that resemble the expected real world conditions in a space application in rate of change and magnitude.

The simulation of the synthetic scenario in Fig. 4 shows an upcoming drift that was configured to fully ramp in at the halfway mark of the simulation time. The difference between the two simulated discipline algorithms can be clearly seen. The SW-PID based algorithm shows a characteristic reset moment at each synchronization interval, while the ALCC algorithm shows a more smooth and continuous correction of the time drift.

The real world derived scenario in Fig. 5 shows the results of the two algorithms in the Eu:Cropis application scenario. Here the temperature delta is shown in addition to the time deviation of the reference clock. It can be seen that the drift rate increases with more severe temperature δ . The mean deviation of the estimated time of the algorithms in this example with the change to a large synchronization interval is at 13.8 ms for the ALCC algorithm and at 139 ms for the SW-PID algorithm which shows an improvement of an order of magnitude.

Simulation runtime depends on the number of clocks simulated, the chosen result interval and the total time simulated. For the examples shown in this paper, a single high resolution run takes about 30 seconds to complete on a modern office workstation. The optimization feature utilizing the GA approach is computationally intense due to the number of simulations required to find a good solution. It is highly

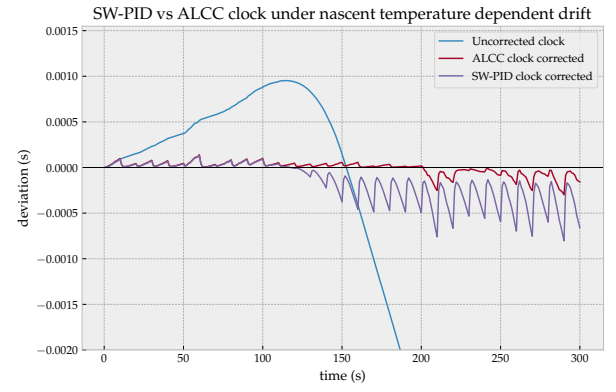


Fig. 4. SW-PID setpoint weighted local clock vs. an ALCC clock under nascent drift conditions. ALCC clock shows significantly better performance as soon as the temperature based drift is introduced

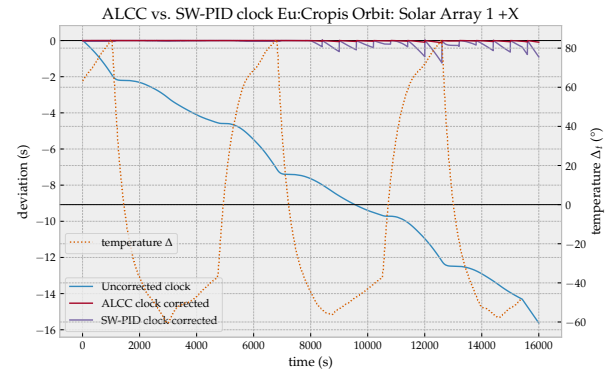


Fig. 5. Synchronization performance of ALCC vs. SW-PID in the Eu:Cropis application scenario at the solar array sensor position

parallelizable but still takes on the order of days on standard office hardware to complete.

VI. CONCLUSION

This paper has presented a novel simulation framework designed specifically for the development and evaluation of local clock discipline algorithms for WSNs in harsh environments. By combining a clock error model with a data-driven event system, the framework provides a reproducible and cost-effective alternative to hardware testing for the initial stages of algorithm design. The case study, using real-world satellite temperature data, demonstrated the framework's ability to clearly differentiate the performance of various algorithms, showing its utility. Furthermore, the integration of a genetic algorithm-based optimization system elevates the framework beyond a simple evaluation tool, enabling the automated discovery of optimal algorithm parameters for specific scenarios.

However, it is important to acknowledge the framework's limitations, which are a direct result of its focused design philosophy. By intentionally isolating the local clock discipline problem, the simulation does not model second-order effects from the time distribution protocol, such as network jitter or packet loss, which can influence the quality of the

synchronization points. Additionally, the clock error model is a behavioral approximation designed to replicate the observable effects of drift, not a complete physical simulation of the underlying crystal oscillator physics. Despite these abstractions, the framework provides a powerful and cost-effective tool for algorithm design, validation, and optimization, significantly de-risking the subsequent, more expensive hardware verification phase.

Future work will focus on expanding the framework's capabilities. This includes integrating models for other environmental factors such as oscillator aging and radiation effects, which are also critical in space applications. Additionally, it is planned to explore the framework coupling with a network-level simulator to investigate the interplay between local clock discipline and time distribution protocols.

REFERENCES

- [1] R.-C. Buta, M. Drobczyk, T. Firchau, A. Luebken, T. P. Palade, A. Pastav, and E. Puschita, "SpaceWire-to-UWB Wireless Interface Units for Intra-spacecraft Communication Links," *Sensors*, vol. 23, no. 3, p. 1363, Jan. 2023.
- [2] R. Amini, G. Aalbers, R. Hamann, W. Jongkind, and P. G. Beethuizen, "New generations of spacecraft data handling systems: Less Harness, More Reliability," in *57th International Astronautical Congress*, ser. International Astronautical Congress (IAF). American Institute of Aeronautics and Astronautics, Oct. 2006.
- [3] S. Li, G. Fan, and G. Xu, "Application of Wireless Satellite Bus in micro-satellite design," in *2009 International Conference on Mechatronics and Automation*, Aug. 2009, pp. 2612–2616.
- [4] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–10.
- [5] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 149–154, Jan. 2003.
- [6] D. Stanislawski, X. Vilajosana, Q. Wang, T. Watteyne, and K. S. J. Pister, "Adaptive Synchronization in IEEE802.15.4e Networks," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 795–802, Feb. 2014.
- [7] B. Martinez, X. Vilajosana, and D. Dujovne, "Accurate Clock Discipline For Long-Term Synchronization Intervals," *IEEE Sensors Journal*, vol. 17, no. 7, pp. 2249–2258, Apr. 2017.
- [8] A. Lübken and A. Förster, "An Adaptive Approach to Time Synchronization for Wireless Sensors Under Extreme Conditions," *IEEE Sensors Journal*, vol. 24, no. 16, pp. 26 592–26 606, Aug. 2024.
- [9] Y. Huang, T. Li, X. Dai, H. Wang, and Y. Yang, "TS2: A realistic IEEE1588 time-synchronization simulator for mobile wireless sensor networks," *SIMULATION*, vol. 91, no. 2, pp. 164–180, Feb. 2015.
- [10] L. Riliskis and E. Osipov, "Symphony: A Framework for Accurate and Holistic WSN Simulation," *Sensors (Basel, Switzerland)*, vol. 15, no. 3, pp. 4677–4699, Feb. 2015.
- [11] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *2007 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 2289–2294.
- [12] H. Y. Kim, "Modeling and Tracking Time-Varying Clock Drifts in Wireless Networks," *PhD Thesis*, 2014.