Energy-efficient Clock-Synchronization in IoT Using Reinforcement Learning

Damir Assylbek, Aizhuldyz Nadirkhanova, Dimitrios Zorbas
Nazarbayev University, School of Engineering & Digital Sciences, Astana, Kazakhstan
Authors' email: {firstname.lastname}@nu.edu.kz

***Abstract*—Clock synchronization in the Internet of Things (IoT) is a critical aspect of ensuring reliable and energy-efficient communications among devices within a network. In this paper, we propose an entirely autonomous and lightweight Reinforcement Learning (RL) approach to learn the periodicity of synchronized beacon transmissions between a transmitter and several receivers, while maximizing the sleep time between successive beacons to conserve energy. To do so, the proposed approach exploits a set of states, actions, and rewards so that each device adapts the radio-on time accordingly. The approach runs on each individual receiver without any prior knowledge of the network status. It is implemented and tested on off-the-shelf ESP32 IoT devices which are known to exhibit high clock drift rates. The testbed results demonstrate the ability of the approach to autonomously synchronize the receivers while achieving a similar performance in terms of packet (beacon) reception ratio but 45% better energy efficiency compared to a traditional approach followed in the literature for one-to-many type of synchronization. Apart from the improved energy consumption, the power characterization of the system shows that the RL approach requires negligible CPU resources.**

***Index Terms*—Machine Learning, Reinforcement Learning, Internet of Things, Synchronization, Wireless networks**

I. Introduction

Clock synchronization in IoT networks is a critical aspect that ensures efficient and reliable communication among networked devices. In IoT and in wireless networks in general, accurate clock synchronization is essential for various applications and aspects of the network, such as data coordination, channel access, and energy consumption [1]. However, achieving precise clock synchronization is challenging due to background process scheduling and the inherent variability in wireless communications, including signal propagation, path-loss, and interference [2].

There are many applications that require one-to-many type of synchronization between devices. In this type of communication, a master device (e.g., a gateway) broadcasts data to multiple end-devices (EDs) in order to synchronize their clock and perform a number of operations without collisions. Typical examples of such an application are the over-the-air firmware updates [3] and time-slotted protocols [4]. Since the EDs need to periodically wake up to receive data, guard times are used between successive beacon receptions in order to tolerate slight clock drifts. However, several issues may arise. Firstly, the transmissions' periodicity must be known to the EDs beforehand. If this changes, the EDs may need to be reprogrammed or updated one by one. Secondly, each ED has its own clock drift due to differences in the crystal oscillator, aging, battery voltage, and operating system scheduling. Achieving precise clock synchronization requires manual adjustment of guard times to each individual ED. Otherwise, long guard times need to be used to compensate for high clock drifts among all devices. Lastly, even if the guard time is perfectly adjusted per ED, clock drift is affected by environmental conditions such as ambient temperature, meaning certain devices may exhibit higher clock drifts. Handling these cases manually may result in significant manpower costs.

To tackle these issues, we propose a RL approach for clock-sleep adjustments on individual devices without prior knowledge of beacon transmission periodicity. This solution uses the SARSA algorithm, defining states, actions, and rewards for full autonomy in clock synchronization and energy savings. Tested on ESP32 devices with IEEE802.11 transceivers, our approach outperformed a deterministic method with fixed guard times. Results show quick adaptation to changes without compromising packet reception ratio. Our approach achieves similar beacon reception success rate, but testing on a power meter revealed average energy savings of 45% per round.

Overall, the contributions of this research are summarized as follows: (a) An autonomous RL approach to learn the beacon periodicity but also adapt to beacon periodicity changes is presented; (b) the proposed approach is capable of adjusting the guard and sleep time of each individual device, minimizing the energy losses; and (c) an open-source C++ implementation for off-the-shelf ESP32 devices is provided.

The rest of the paper is structured as follows. Section II surveys the related work in the area of time synchronization in IoT networks. Section III presents the proposed solution for the one-to-many synchronization scheme. The evaluation of the proposed solution using real field experiments is presented in Section IV. Finally, Section V draws the conclusions and ideas for future work.

II. Related Research

Learning approaches in IoT for clock synchronization blend conventional and machine learning (ML) techniques to automate tasks and address challenges. Traditional solutions focused on synchronizing resource-constrained devices with minimal complexity. The Tiny-Sync protocol [5] offers a straightforward method by estimating clock drift upon receiving a packet, provided there's no sleep interval between beacons. Slot-based synchronization methods, like 6TiSCH [6], ensure receivers wake up just in time for transmissions, synchronizing by the duration the radio was active until beacon receipt. A similar strategy is applied in [4] for one-to-many synchronization by averaging sleep times based on beacon receptions. Magzym et al. [7] introduce a deep sleep mode post-synchronization, assuming a known and fixed beacon periodicity and a long guard time to account for the maximum clock drift.

Fan et al. [8] highlight the limitations of traditional clock-synchronization methods, such as the Huygens algorithm [9], which may require complex hardware not always available in IoT devices. They critique the high communication overhead and internet dependency of protocols like NTP or the Multi-hop Precision Time Protocol (M-PTP) [10]. Instead, they propose a lightweight, probe-based synchronization approach achieving accuracy within tens of microseconds for devices with low clock drift.

Similar to this, a method [11] utilizes linear regression to model clock drift, facilitating mutual device synchronization with considerable precision. The Flooding Time Synchronization Protocol (FTSP) [12] and the Rate-Adaptive Time Synchronization (RATS) [13] also employ linear regression to average timestamps for final clock adjustment and to minimize the confidence interval for time estimation, respectively. However, these methods face challenges in multi-device scenarios due to coordination complexities.

Abakasanga et al. [14] propose an unsupervised deep learning framework for synchronization in multi-hop networks, leveraging signal characteristics and propagation delays. This innovative approach simplifies deployment by training directly on the devices, although it's not suited for environments where beacon periodicity is unknown to the devices.

III. RL-based Clock Synchronization

In this section, we present a SARSA-based RL solution to autonomously adapt the sleep time of the end-devices and compensate for changes in the clock drift. We first explain the examined scenario, some SARSA fundamentals, and then, we present the clock-synchronization process.

We examine the scenario where one transmitter (master) periodically transmits packets (beacons) to simultaneously synchronize several ED receivers (slaves) using a single beacon. The master's clock (time of the transmission) is used as a reference clock for the slave devices. The problem is challenging because both the master and the slave devices may drift over time. That means that the beacon periodicity may not be very precise but slaves have to adapt their guard times accordingly.

It is assumed that the receivers are in sleep mode for the entire period and wake up only to receive the transmitter's beacon. The objective of the approach, which independently runs on each of the receivers, is threefold: (a) to quickly learn the time interval between these successive periodic transmissions without any prior knowledge of the network characteristics or the environmental conditions; (b) to autonomously achieve a high synchronization accuracy which may be different per ED due to different environmental conditions and crystal characteristics; and (c) to quickly adapt to network changes and minimize desynchronizations.

Let us denote with d the time interval between two successive beacon transmissions initiated by a master device. No prior information about d is known by the EDs. We assume that an ED i is synchronized when it receives a beacon within a time limit (see initial guard time threshold below) from the time it wakes up. However, the algorithm may increase that time if beacons are missed. The amount of time an ED i spends waiting until it receives a beacon is denoted with δ_i. The value of δ_i and whether it is withing the guard time threshold is used to decide about future actions as well as about the sleep time for the next iteration denoted with l_i.

An ED i is considered synchronized only when it has successfully calibrated l_i while minimizing the likelihood of missed beacons or timeouts by adjusting the guard time. Particularly, δ_i and timeout threshold serve as important parameters that guide these adjustments, so that each ED i stays synchronized.

Waking up too early or too late (see Radio on timeout below) can lead to a missing beacon, and perhaps to a desynchronization. An ED i considers itself "desynchronized" when it misses the beacon for three successive iterations. Therefore, each ED i employs SARSA to dynamically select an action to adjust its guard time and its sleep time based on the outcomes of previous beacon receptions. By following actions and selecting rewards for a set of states, an ED i first discovers a rough interval between two successive beacons d, and next, it fine-tunes l_i to maximize its sleep time until the next wake-up.

To capture the status of the beacon reception, two significant threshold values are defined in the system as follows:
Initial Guard time threshold: This is a time buffer set to ensure that each ED i wakes up earlier than the expected beacon transmission so that it has enough time to turn on its radio and listen for incoming beacons. The guard time compensates for accumulated drift time between two synchronization iterations as well as scheduling changes. For example, setting this threshold to 10ms means that the ED will wake up 10ms before the expected arrival time of the beacon. If δ_i is less than the guard time, the ED i assumes that it is well-synchronized, however, it will continue to make micro-adjustments in the guard time up to a point where it does not miss beacons and, at the same time, δ_i gets as low as possible. This is assumed to be the perfect synchronization and suggests that current Q-values are effective, so no SARSA learning update is required anymore.
Maximum Guard time threshold (Optional): This threshold determines the maximum value that the guard time can reach. The purpose of this threshold is to prevent ED from ending-up with a very high guard time value due to successive beacon misses that could lead to a very high energy consumption. In practice, it can provide a trade-off between beacon loss and energy consumption for super lossy and high clock-drift devices.
Radio-on timeout: This threshold determines the maximum amount of time the radio should be on while waiting for a beacon before going back to sleep. If the beacon is not received

TABLE I
STATES (S), ACTIONS (A), AND REWARDS (R) OF THE APPROACH.

States	Actions (ms)	Rewards
0	0 (radio on/off)	None
1	+1, +2, +3, +5, +10	0, 1
2	-1, -2, -3, -5, -10	0, 1
3	-2, -1, +1, +2	0, 1
4	+1, +2	0, 1
5	0	None

within this time limit, it is assumed that something may be wrong with the synchronization, thus, the SARSA process is triggered to adjust l_i in order to better align with the beacon timing in future iterations. However, since beacons may also be missed because of the path-loss, SARSA handles these cases conservatively.

Table III describes the states, actions, and rewards of the approach. The system is designed with six possible states, each associated with a set of actions defined to adjust l_i:

State 0 (Initial calibration phase/Re-calibration): When the device is first powered up it enters the initial calibration phase, denoted as State 0. During this phase, the primary objective is to discover d, which is crucial for synchronization. The process of learning d is done through an exclusive action, referred to as Action 0 (switching the radio on/off). If d is unknown, Action 0 allows the ED to constantly have each radio on for two consecutive beacons. Once this happens, the state of the ED changes, indicating the initial calibration is complete. If the beacon transmission period d has unexpectedly changed and ED has missed three consecutive beacons (i.e. 3 consecutive radio-on timeouts), it will return to the State 0 and begin the re-calibration process.

State 1 (Beacon received too early and outside the guard time threshold): State 1 signifies that the ED has received a beacon, but the arrival time exceeds the current guard time threshold (i.e., high δ_i). This may happen when the ED wakes up too early in relation to the beacon transmission. In this state, the device employs actions designed to increase l_i, thus, increasing the sleep time in future iterations.

State 2 (Beacon received but not close to the middle of the guard time threshold): State 2 is specifically designed for scenarios where the ED achieves synchronization (the beacon is received within the guard time threshold or before the radio-on timeout), but fine-tuning is required to achieve a synchronization close to the middle of the guard time. Shorter actions could be chosen but this would increase the convergence time of the approach.

State 3 (Beacon missed – it was not received before the radio-on timeout): State 3 signifies that the ED i has failed to receive the beacon within the radio-on timeout threshold. In this state, the device employs actions designed to reduce l_i, thus, waking up earlier in future iterations. This adjustment aims to correct δ_i for ED i to improve the alignment with future beacon transmissions.

State 4 (Beacon missed – Guard time threshold adjustment): State 4 comes after State 3 if a beacon is missed

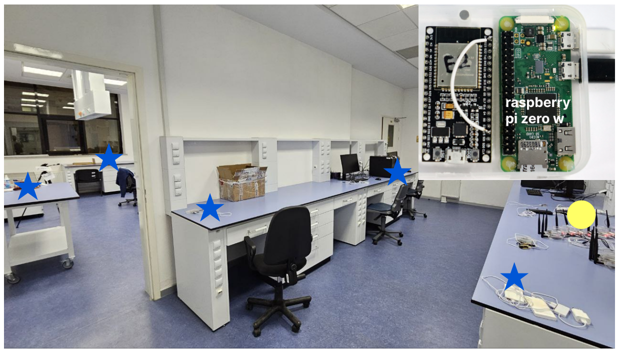

Fig. 1. Setup of the experiments indicating 1 master device (circle) and 5 slave device (stars). At the upper right corner, an ESP32-WROOM ED connected to a Raspberry Pi Zero is shown to capture the ED's serial output.

which may be caused because of the too-short selected guard time. The guard time will not be increased if it exceeds the maximum guard time threshold (if the latter has been set in the system). Using this action, the algorithm can find a guard time limit at which the ED does not miss beacons due to bad synchronization. To do so, the initial guard time must be set to a low value which will gradually be increased until the limit is reached. Another approach would be to start with a high value of initial guard time and decrease its value every time the ED successfully receives a beacon in the middle of the guard time. However, this scenario would require more steps to converge and one more action to revert to the previous guard time state once the device starts missing beacons.

State 5 (Perfect Synchronization): In this state, we assume that an ED is perfectly synchronized when δ_i is very close to the middle of the current guard time (+/- 1ms) and we do not miss beacons. No actions are required in this case.

In SARSA, States 1, 2, 3, and 4 use a binary reward structure where a reward of 1 indicates an improvement of δ_i, and a reward of 0 indicates no improvement or a missed beacon. This binary approach simplifies the learning process, allowing the SARSA algorithm to quickly adjust l_i and refine the synchronization performance.

Furthermore, as ED i adjusts its exploration rate and the number of received and missed beacons, SARSA transits from exploration to exploitation. This gradual shift helps it to maintain an efficient sleep time.

IV. EVALUATION & DISCUSSION OF THE RESULTS

To evaluate the proposed approach, a series of experiments were conducted with 1 master and 5 slave devices randomly placed in a $40m^2$ room (see Fig. 1). The solution was implemented on ESP32 devices such as ESP32-WROOM and Lilygo TTGO V2 using the C++ programming language [1]. IEEE802.11 transceivers and a connection-less protocol called ESP-NOW were employed for the tests. The solution was compared to another synchronized approach [7] where the interval between two successive beacon transmissions (i.e., d) was already known to the EDs and fixed throughout the

[1]https://github.com/nu-iot-lab/RL-Clock-Drift-Correction

TABLE II
EXPERIMENT PARAMETERS & SETTINGS

Parameter	Value
Master/Slave devices	1/5
PHY	ESP-NOW (IEEE802.11)
Data rate	250 Kbps
Tx power	20 dBm
Payload length	8 Bytes
Beacon transmission interval (d)	5, 10, 30 seconds
Initial Guard time threshold	10 ms
Maximum Guard time threshold	35 ms
Radio-on timeout	35 ms
Reinforcement Learning parameters	
Learning rate (α)	0.1
Exploration rate (ϵ)	1.0
Minimum exploration rate (ϵ_{min})	0.05
Epsilon decay (ϵ_{decay})	0.001
Discount Factor (γ)	1

process. The average Received Signal Strength among devices was -71dBm with a minimum of -88dBm. The misses due the path-loss were negligible.

Three scenarios with different time intervals between beacons were tested. Each scenario was executed 10 times and the average results as well as the standard deviation of these results are presented. We measured the Packet Reception Ratio (PRR), the convergence time of the RL approach as well as its stability. "Convergence" is defined as the time needed (expressed in beacon rounds) to achieve a synchronization without losses. It actually reveals how fast the EDs learn d and fine-tune their guard and sleep time. "Stability" is defined as the maximum δ values for all beacon rounds. Because in ESP32 devices the wake-up time differs several milliseconds per round (mainly due to the background RTOS scheduling), continuous sleep time corrections are usually required. Stability expresses the adaptation of the approaches to fluctuations in wake-up times.

In the SARSA algorithm, we have adopted standard parameter settings as outlined in Table II. A learning rate of 0.1 is selected to balance between the exploration and the exploitation. The exploration rate is set to 1.0 to ensure that the algorithm goes through all possible actions, which is crucial at the early stages of the learning process in order to learn the periodicity of the beacon transmissions. If it was set to a lower value, SARSA would start exploiting the Q-table earlier, which could prevent it from discovering potentially better options.

The minimum exploration rate of 0.05 maintains the baseline level of exploration, preventing SARSA from becoming too rigid. An epsilon decay of 0.001 means that the exploration rate will decrease very slowly with each step and ensure a prolonged period of exploration. However, the discount factor (γ) was set to 1 so that our approach adjusts l and converges as quickly as possible. This means that future rewards will be treated the same as immediate rewards. For future rewards, the discount factor will not be applied at all and SARSA will treat all actions equally important. All these parameters together guide SARSA to allow exploration in the beginning and gradually transfer to exploitation.

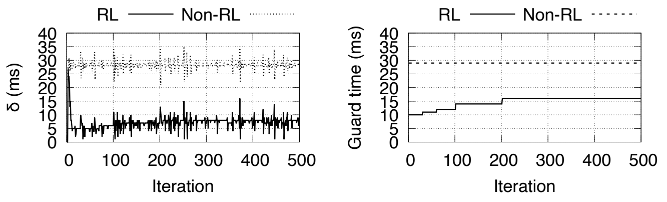

Fig. 2. (left) Value of δ throughout a random instance for both approaches and d=30sec (0 = beacon missed); (right) Value of guard time throughout a random instance. The RL approach calibrates the guard time to a higher value to avoid beacon misses.

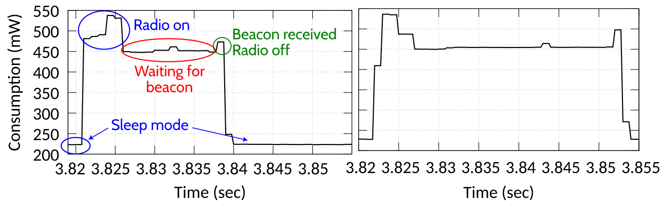

Fig. 3. Power consumption comparison between the RL approach (upper) and the non-RL approach (lower) for a random iteration.

Tables III and IV present the average results of the experiments for the two approaches, respectively. The time intervals between transmissions is equal to 5, 10, and 30 seconds. It can be observed that the RL approach achieved a very high PRR and very similar to the hardcoded solution. Indeed, in all experiments, all devices successfully synchronized with the transmitter and missed maximum 4 out of 500 beacons after convergence. The initial synchronization was done within 2 beacons while the fine-tuning took another 10 to 11 beacons (iterations) to converge to a fine-tuned synchronization. Despite the huge clock-drift nature of ESP32 devices, the approach showed remarkable stability with only 7-9ms average difference between the minimum and the maximum δ times. The proposed solution achieved an even lower average δ time compared to the non-RL approach. This is also shown in Fig. 2(left) for a randomly chosen ED. The non-RL approach could not achieve a δ value below 22ms. On the contrary, the RL approach quickly reduces the variability in δ after the calibration phase while achieving an average δ of 9ms. Four misses occurred throughout the process as it can be seen from Fig. 2(right). The approach reacted to those misses with a slight increase in guard time. As it was stated previously, the ED starts with a low initial guard time and gradually increases this value for every miss until it find a guard time limit.

The guard time in the non-RL approach was fixed to 29ms for all devices which was the maximum recorded value among RL devices in the experiments. On the contrary, the ED equipped with the RL approach could achieve a 62% lower guard time (i.e., less waiting time) compared to the non-RL approach. In terms of energy savings, this is translated to an average of 9.5mJ less energy consumption (45% overall improvement) in each round due to the reduced radio-on time compared to the non-RL solution throughout the process. This finding is confirmed by Fig. 3, where two devices – running the RL and the non-RL approach, respectively – were deployed on a power meter and a random iteration was used to compare their power consumption. We can observe from this figure that

TABLE III
RESULTS OF THE RL APPROACH FOR VARIABLE BEACON TRANSMISSION INTERVALS d.

d (sec)	overall PRR (%)		PRR after stabilizing the guard time (%)		Convergence (beacons)		Stability (ms)		δ (ms)		Guard time (ms)			
	Avg	σ	Avg	σ	Avg	σ	Avg	σ	Avg	σ	Avg	σ	Min	Max
5	99.16	0.43	99.52	0.86	10.12	2.47	7.24	2.49	7.25	2.25	17.6	5.87	11	29
10	98.64	0.83	99.05	1.27	9.76	2.68	7.68	2.81	7.64	3.06	18.6	6.14	12	29
30	99.0	0.24	99.68	0.33	10.88	4.64	9.08	2.94	8.52	2.65	21.2	6.13	13	29

TABLE IV
RESULTS OF THE NON-RL APPROACH FOR VARIABLE BEACON TRANSMISSION INTERVALS d.

d (sec)	PRR %		Stability (ms)		δ (ms)		Guard time (ms)
	Avg	σ	Avg	σ	Avg	σ	Fixed
5	99.65	0.19	13.25	6.08	27.81	0.01	29
10	99.08	1.1	11.4	5.08	27.85	0.01	
30	99.56	0.44	12.2	2.49	28.05	0.02	

the "waiting for beacon" time is much longer in the non-RL approach with a fixed guard time. The SARSA computation takes place once the beacon is received (or missed), however, its execution is negligible to be shown in the figure ($\sim$85μs long on average).

V. CONCLUSION & FUTURE RESEARCH

In this paper, we examined the possibility of autonomously adapting the guard and sleep times of multiple receivers which get synchronized by periodically receiving beacons from a master device. A very low cost Reinforcement Learning approach based on SARSA was proposed and evaluated using a testbed consisting of ESP32 devices. The approach achieved an average of 62% lower radio-on time compared to a non-RL approach and 50% less energy consumption.

In the future, we are planning to implement an efficient transfer learning approach so that new registered neighboring devices can accelerate their learning process and adapt their sleep times accordingly.

ACKNOWLEDGEMENT

This publication has emanated from research conducted with the financial support of Nazarbayev University grant No. 11022021FD2916 for the project "DELITMENT: DEterministic Long-range IoT MEsh NeTworks".

REFERENCES

[1] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad hoc networks*, vol. 3, no. 3, pp. 281–323, 2005.

[2] D. Capriglione, D. Casinelli, and L. Ferrigno, "Analysis of quantities influencing the performance of time synchronization based on linear regression in low cost WSNs," *Measurement*, vol. 77, pp. 105–116, 2016.

[3] K. Abdelfadeel, T. Farrell, D. McDonald, and D. Pesch, "How to Make Firmware Updates over LoRaWAN Possible," in *21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pp. 16–25, 2020.

[4] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things," *Computer Communications*, vol. 153, pp. 1 – 10, 2020.

[5] S. Yoon, C. Veerarittiphan, and M. L. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 2, pp. 8–es, 2007.

[6] T. Chang, T. Watteyne, K. Pister, and Q. Wang, "Adaptive synchronization in multi-hop TSCH networks," *Computer Networks*, vol. 76, pp. 165–176, 2015.

[7] Y. Magzym, A. Eduard, D. Urazayev, X. Fafoutis, and D. Zorbas, "Synchronized ESP-NOW for Improved Energy Efficiency," in *11th International Black Sea Conference on Communications and Networking*, IEEE, 2023.

[8] Z. Fan, Y. Xu, P. Liu, X. Li, R. Zhang, T. Yang, W. Wu, Y. Li, L. Chen, and G. Zhang, "SSA: Microsecond-Level Clock Synchronization Based on Machine Learning for IoT Devices," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–12, 2023.

[9] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 81–94, 2018.

[10] K. He, C. An, J. H. Wang, T. Li, L. Zu, and F. Li, "Multi-hop Precision Time Protocol: an Internet Applicable Time Synchronization Scheme," in *IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2022.

[11] J. J. Pérez-Solano and S. Felici-Castell, "Adaptive time window linear regression algorithm for accurate time synchronization in wireless sensor networks," *Ad Hoc Networks*, vol. 24, pp. 92–108, 2015.

[12] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 39–49, 2004.

[13] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M. B. Srivastava, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 130–141, 2005.

[14] E. Abakasanga, N. Shlezinger, and R. Dabora, "Unsupervised Deep-Learning for Distributed Clock Synchronization in Wireless Networks," *IEEE Transactions on Vehicular Technology*, 2023.