Time-Sensitive Networking for Trajectory Tracking of an Unmanned Ground Vehicle over Wi-Fi

Elena Ferrari
Department of Information Engineering
University of Padova
Padova, Italy
elena.ferrari.7@phd.unipd.it

Dave Cavalcanti
Intel Labs
Intel Corporation
Hillsboro, Oregon, USA
dave.cavalcanti@intel.com

Valerio Frascolla
Intel Deutschland GmbH
Intel Corporation
Munich, Germany
valerio.frascolla@intel.com

Susruth Sudhakaran
Intel Labs
Intel Corporation
Hillsboro, Oregon, USA
susruth.sudhakaran@intel.com

Alberto Morato
National Research Council of Italy
CNR-IEIIT
Padova, Italy
alberto.morato@cnr.it

Stefano Vitturi
National Research Council of Italy
CNR-IEIIT
Padova, Italy
stefano.vitturi@cnr.it

Angelo Cenedese
Department of Information Engineering &
Department of Industrial Engineering
University of Padova
Padova, Italy
angelo.cenedese@unipd.it

***Abstract*—The demand for precise time synchronization is of great interest in contemporary networked systems, especially in the context of converged networks where seamless communication among devices, actuators, and sensors is imperative. This has led to the development and adoption of Time-Sensitive Networking (TSN) and Wireless TSN (WTSN) technologies, with a particular attention to the IEEE 802.1AS Generalized Precision Time Protocol (gPTP) standard. In this work, we explore the role of time synchronization in a wireless network scenario involving an Unmanned Ground Vehicle (UGV) that has to track a desired time dependent trajectory. The UGV receives trajectory waypoints from a secondary station, emphasizing the importance of precise timing in trajectory tracking. Utilizing the IEEE 802.1AS standard for wireless time synchronization, this study investigates the impact of clock synchronization errors and latency on trajectory tracking. Hence, it demonstrates the capability and benefits of IEEE 802.1AS in managing device clocks and facilitating time correction to ensure precise trajectory tracking despite synchronization errors and latency.**

***Index Terms*—Time synchronization, Wireless communication, IEEE 802.1AS, UGV, TSN, WTSN**

I. Introduction

With the advent of Industrial Internet of Things (IIoT) there is a growing interest in creating converged unified networks since these offer the scalability and flexibility needed in IIoT applications [1]. To address the still unsolved challenges of such networks and to create more integrated, predictable, and standards-based networks, Time-Sensitive Networking (TSN) and Wireless TSN (WTSN) have been developed [2]–[4]. TSN capabilities are defined as part of the family of IEEE 802.1 Standards, the goal of which is to enable converged networks by ensuring time-synchronization, determinism, low latency, low jitter, and reliability for time-critical communications, also in conjunction with cellular networks [5]. The research community has recently become rather active on the IEEE 802.1AS and the WTSN topics; for instance, authors in [6] elaborate on how Ethernet TSN and WTSN can synergize and provide promising results in an industrial control system. Authors in [7] focus instead on a communication-control co-design approach targeting use cases with stringent latency requirements when steering robots in IIoT scenarios.

In this work, we give particular attention to IEEE 802.1AS, which can operate over wired and wireless links and is a level-2 profile of the IEEE 1588 Principal Time Protocol (PTP), which ensures synchronized time transport, source selection, and notification of impairments such as phase and frequency discontinuities [8], [9]. Time synchronization allows all the devices in a converged network to maintain consistent time for recording transactions. Timestamps can be also used to coordinate actions among distributed devices like sensors, actuators, and controllers, ultimately enhancing process efficiency.

Time synchronization is a TSN core capability playing a central role in many fields such as data centers, financial systems, telecommunications, and especially in manufacturing systems based on IIoT [10]. One of the most popular applications in the context of IIoT that could get significant advantages from the introduction of time synchronization is mobile robotics [11] and, in particular, applications that involve the use of Unmanned Ground Vehicles (UGVs). As a matter of fact, when dealing with mobile robots, the choice of employing WTSN is necessary to enable the UGV to freely move in the environment without limitations due to wired cabling. Clearly, in this context, the opportunity to achieve time synchronization among mobile devices is particularly appealing to improve the performance figures of the addressed applications. Thus, to delve into the time synchronization capabilities of WTSN,

This study was carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043), and was also partially funded by the European Commission Horizon Europe SNS JU PREDICT-6G (GA 101095890) Project. This manuscript reflects only the Authors' views and opinions, the EC can not be considered responsible for them.

we focus on an application in which a UGV has to follow a trajectory that is dynamically transmitted by another device via Wi-Fi. We focus on Wi-Fi among the different wireless technologies specifically to employ the TSN feature, thanks to the significant enhancements in the latest versions of the standard [12]. The study is based on realistic simulations using parameters obtained from practical experiments executed using real testbeds. Specifically, we focus not only on the benefits brought by time synchronization but also on the possible drawbacks that could arise from synchronization errors. In this direction, we examine different scenarios where various synchronization errors are injected, including constant and drifting offsets, i.e. asymmetry in the synchronization of the clocks. Finally, we explore methods to mitigate the effects of such errors on the performance of the UGV application.

The paper is structured as follows. Section 2 provides an overview of the concept of time synchronization, highlighting the relevance of the IEEE 802.1AS standard. Section 3 describes the experimental setup and Section 4 presents and discusses the outcomes of the experimental sessions. Section 5 concludes the paper.

II. Background and Motivation

In the last two decades, the significance of time synchronization has surged, emerging as a key infrastructure for distributed systems, such as environmental monitoring, data fusion, autonomous driving, and power management [8], [13], [14]. While existing literature extensively addresses the time synchronization issue, the introduction of IEEE 802.1AS standard has revolutionized synchronization accuracy, achieving in some cases nanosecond precision, a notable enhancement compared to the previous methodologies as, for example, Network Time Protocol (NTP) [15] and Simple Network Time Protocol (SNTP) [14], which ensure a millisecond precision [16]. The IEEE 802.1AS is a core capability within the TSN domain for wired and wireless, which underscores its central role in contemporary real-time networked systems.

Studies have showcased the applicability of time synchronization in diverse scenarios as discussed in [17] and [18]. In [17], which highlights the capabilities of WTSN within a collaborative robotic workcell comprising two robotic arms, simulating a material handling scenario, the focus lies on exploring various configurations and devising measurement methodologies to correlate wireless network performance. Instead, in [18], the performed analysis focuses again on a collaborative task, detailing a methodology to align the Quality of Service (QoS) requirements from the application layer of the Robotic Operating System 2 (ROS2) and Data Distribution Service (DDS) middleware with the link layer transport utilizing WTSN. Differently from these works, we emphasize the application of IEEE 802.1AS over a simulated mobile robotic scenario. We delve into the intricacies of managing time-related offsets and errors in time synchronization, highlighting its critical role in ensuring precision control of a UGV. In particular, our investigation extends beyond mere implementation, delving into the nuanced effects of synchronization error on time-dependent systems. Through meticulous analysis, we elucidate how precise time synchronization facilitates timely interventions, allowing to effectively adjust to meet the demands of various tasks. Hence, our study aims to contribute to the broader understanding of time synchronization in contemporary distributed systems, such as the ones involving mobile robotics.

III. Experiments Setup

A. Communication architecture

The system architecture consists of two Wi-Fi stations (STAs), as shown in Fig. 1. The first STA generates the desired trajectory to be tracked and simulates the transmission process to the Slave via User Datagram Protocol (UDP). Moreover, the second STA receives the packets and computes the suited linear and angular velocities to reach the specified waypoint sent in the just received UDP frame. From a time synchronization point of view, the waypoint generator (WG) also acts as a Grand Master (GM) while the UGV works as a Slave, as defined in the IEEE 802.1AS standard.

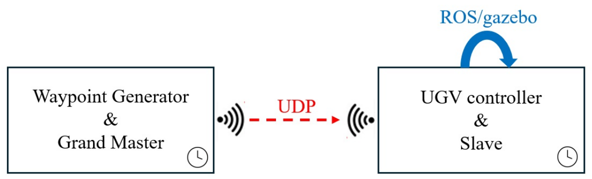


Fig. 1: Scheme of the employed architecture.

Each transmitted UDP frame consists of 256 Bytes, including the following fields.

- header, including sender and receiver device address, UDP frame length, and checksum
- padding, consisting of four Bytes set to zero to prevent protocol encoding errors
- timestamp
- packet number
- x desired position, defined in the World Frame $\mathcal{F}_W$
- y desired position, defined in the World Frame $\mathcal{F}_W$
- θ desired heading angle, defined in the World Frame $\mathcal{F}_W$
- actual time, corresponding to the sending time instant
- desired time for the UGV to reach (x, y, θ), calculated by adding $100\mu s$ to the actual time.

The Slave receives UDP frames and computes the suited linear and angular velocities to navigate toward the designated waypoint provided by the GM. Subsequently, the velocities values are published to the appropriate ROS topic.

B. Waypoints trajectory generation

The intended trajectory for the UGV is achieved through a series of waypoints, ensuring that it reaches the desired pose (x, y, θ) at a specific and predetermined target time, denoted as t_{des}.

The WG generates the trajectory according to the shape outlined in Eq. (1),

$$\begin{aligned} x &= \sin(\omega t) \\ y &= \sin(2\omega t) \\ \theta &= tan^{-1}\left(\frac{dx(t)}{dy(t)}\right) \end{aligned} \tag{1}$$

where $\omega = 2\pi f$ with $f = 1s^{-1}$ and $t = 10000 \cdot pkg_{ID} \cdot 10^{-9}$, which is expressed in seconds and depends on the UDP frame number, i.e., pkg_{ID}. The resulting desired trajectory to be tracked is shown in Fig. 2. This has been segmented into 10000 waypoints, with the WG sending a UDP frame transmission every $10ms$, so that $t_{des} = t_{send} + 10000000$ ns.

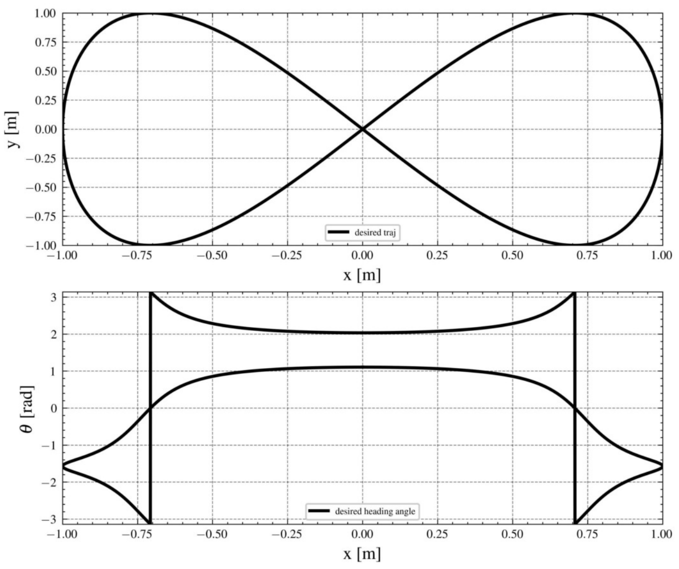


Fig. 2: Desired UGV trajectory to track and desired heading angle.

C. Time synchronization offset addition

To underscore the importance of time synchronization, we conduct tests involving various offsets between the GM and Slave clocks to assess their impact on the accuracy of trajectory tracking. In fact, by leveraging time synchronization, both the sending and the receiving timestamps are known. Hence, to introduce a synchronization error between the two clocks, we introduce two types of offsets: offsets manually imposed by us and offsets extracted from real acquired values. Firstly, we augment with different constant offsets, specifically $1\mu s$, $10\mu s$, $100\mu s$, and $1ms$, the clock of the receiving device. Such clock variations simulate a consistent synchronization error between the involved devices, accounting for potential differences in the internal mechanisms, such as variations in their crystal oscillators or software offsets in the time synchronization.

Then, two divergent offsets are added, in agreement with the fact that their values are directly proportional to the packet number. In such a way, we proportionally increase the offset by $1\mu s$ first and $10\mu s$ later with respect to the packet number. This choice is motivated by the fact that clock synchronization can drift due to various factors such as hardware imperfections, temperature fluctuations, malicious attacks, or the absence of time synchronization. These factors can affect the clock, accumulating error over time and observing in the end a significant discrepancy between the clocks of the involved devices.

To model a real-world scenario, we utilize the acquired data related to the error on time synchronization over Wi-Fi 6 operating over 5GHz between two devices, in detail two Next Unit of Computing (NUC) equipped with an Intel processor (i5-1135G7) running Ubuntu 20.04. Observing the density distribution of the acquired data, shown in Fig. 3, we fit them as a Gaussian Probability Density Function (PDF) indicated by the red line in Fig. 3.

To simulate this realistic time error, we add a random value from this distribution to the receiver clock upon each receipt of a new UDP frame. The randomly generated added time errors are depicted in green in Fig. 3, where they are compared with the previously acquired real data.

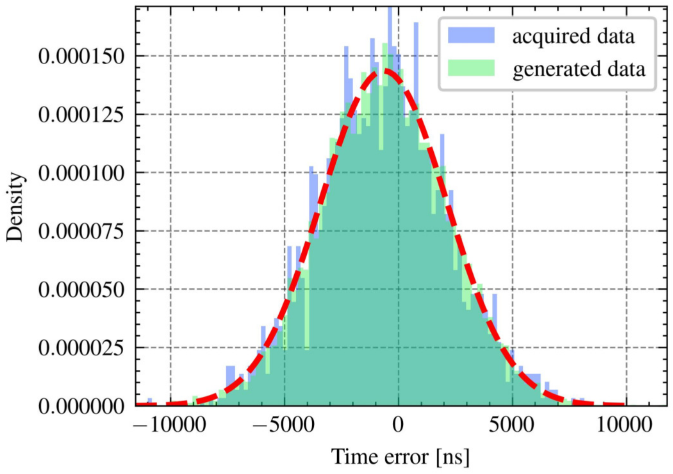


Fig. 3: In blue it is shown the density of the acquired data related to the error on time synchronization in a real scenario, while in red it is given the fitted Gaussian PDF based on the real data. In green, instead, the random data generated accordingly with the known Gaussian PDF is shown.

Furthermore, we take into account also the latency in the communication that would occur in a real-world scenario. The latency data utilized are related to 256 Bytes UDP packets and are acquired by measuring the latency between the two NUCs communicating over Wi-Fi 6 and operating over 5GHz without TSN enhancements. Leveraging these measures, we construct the suited distribution utilizing the *netem* and *tc qdisc* tools. The histogram describing the distribution of these acquired data is illustrated in Fig. 4.

D. UGV controller

With the implementation of time synchronization between the two devices, the receiver can precisely determine both the

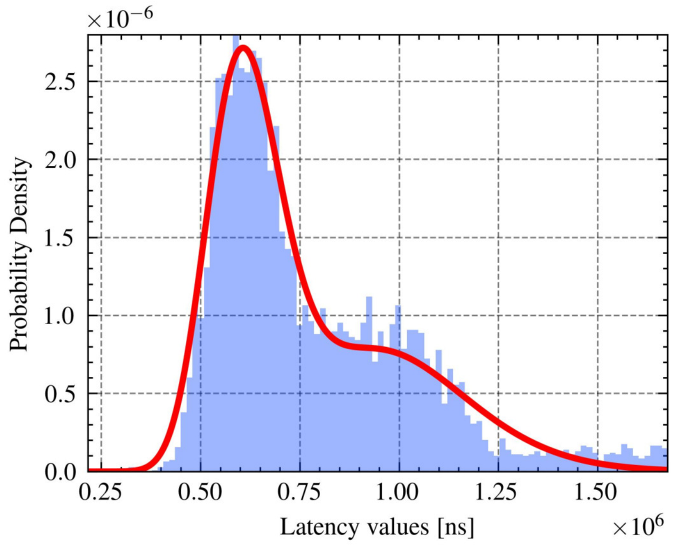


Fig. 4: Histogram illustrating the density of the latency measures acquired in a real scenario, which are fitted by a PDF distribution described as the weighted sum of two gamma distributions, shown with the red line.

sending and the receiving time instants relevant to the same clock. In a first scenario, the controller given in Fig. 5 was employed to compute the suited linear and angular velocities on the receiver side.

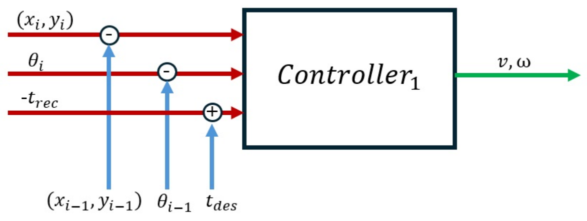


Fig. 5: Input and output variables of the controller adopted to correct the offsets between the involved clocks.

In this configuration, the controller knows the Euclidean distance and the error between the heading angle of the next and the last waypoints, (x_i, y_i, θ_i) and $(x_{i-1}, y_{i-1}, \theta_{i-1})$, respectively. Additionally, the time duration to reach the next waypoint, i.e. τ, can be computed as in Eq. (2), thus underlying the relevance of time synchronization:

$$\tau = t_{des} - t_{rec} \tag{2}$$

where t_{des} is the desired time computed by the sender/GM, while t_{rec} indicates the time instant at which the UDP frame containing the next waypoint is received by the Slave.

Hence, the linear and angular velocities, i.e., v and ω respectively, can be computed as provided in Eq. (3):

$$v = \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{\tau} \qquad \omega = \frac{\theta_i - \theta_{i-1}}{\tau} \tag{3}$$

To mimic a real system we add some latency in wireless communication between devices, since it introduces a transmission delay, slowing down the transmission of packets. Therefore, since this delay can be considerable, possibly causing packets to be lost, time synchronization makes it possible to know the timestamp at which the packet was sent and, therefore, calculate the appropriate speed as in Eq. (4). In this case, the velocities values can be computed taking into account also t_{send}, differently from the previous situation where just t_{des} and t_{rec} are considered, to compensate for the latency effects. The inputs of the controller and outputs are given in Fig. 6, according to

$$v = \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{\tau_2} \qquad \omega = \frac{\theta_i - \theta_{i-1}}{\tau_2} \tag{4}$$

where $\tau_2 = t_{des} + \Delta t - t_{rec}$, with $\Delta t = t_{rec} - t_{send}$ describes the time required to transmit the UDP frame.

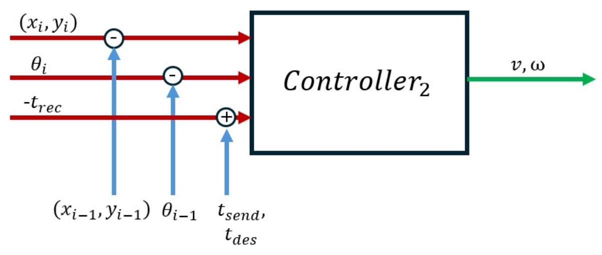


Fig. 6: Input and output variables of the controller adopted to correct the latency.

IV. Results

This section delves into the results obtained in the trajectory tracking scenario. Specifically, five distinct cases have been investigated:

1) Neither offset on the receiver clock device nor latency in the communication channel is added.
2) Constant offset on UGV clock is added. Various constant offset values, including $1\mu s$, $10\mu s$, $100\mu s$, and $1ms$, have been examined on the UGV clock.
3) Random offset from real experimental data is considered. A random offset value, drawn from a distribution representing the typical offset between two clocks in real-world tests under similar conditions, has been applied to the UGV clock, accordingly with the distribution provided in Fig. 3.
4) An increasing offset value, proportional to the UDP frame number, is added to the UGV clock. Two tests have been performed. The first one adopting an offset of $1\mu s$ and the second using an offset of $10\mu s$. Both of them are multiplied by the UDP frame number to create a direct proportionality between offset and UDP frame number.
5) Latency is included. Latency has been simulated based on the real experimental data shown in Fig. 4.

In each of these scenarios, the UDP frames are transmitted from the waypoint generator to the UGV with a period of $10ms$, matching the duration added to the sending time instant to represent the desired time t_{des}.

A. First Scenario

In the first scenario, where no offset is introduced in the UGV clock relative to that of the waypoint generator, the tracked trajectory is shown in Fig. 7

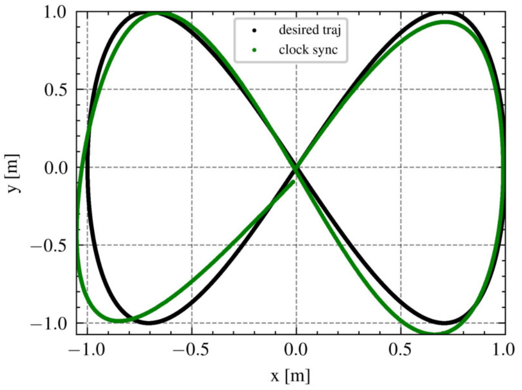


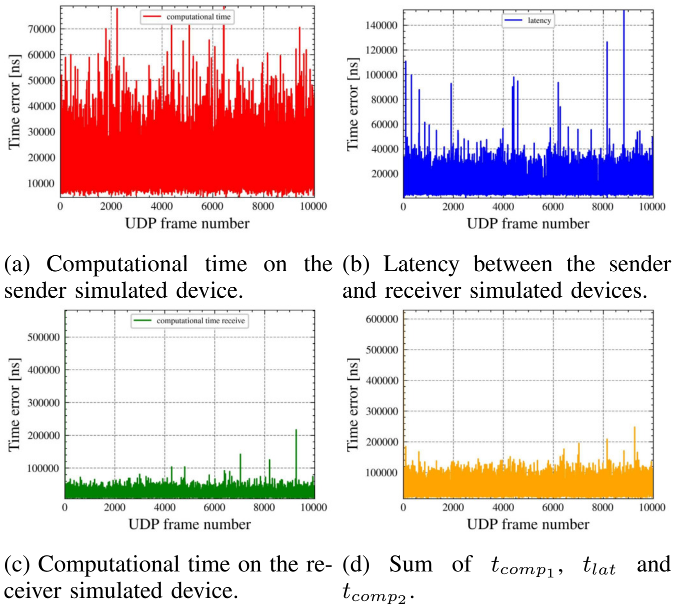
Fig. 7: Trajectory tracking in the scenario where neither offset nor latency between the two clocks is added.

When comparing the tracked trajectories with the desired trajectory, which evolves over time, several factors come into play. Along with latency (t_{lat}), defined as End-to-End (E2E) latency, also the computational time on both the two STAs, i.e. t_{comp_1} and t_{comp_2}, must be considered. In the waypoint generator, the computational time has been computed as the difference between the timestamp at which it sends the UDP frame (t_{send}) and the timestamp at which it starts to generate the waypoint (t_{start}), i.e. $t_{comp_1} = t_{send} - t_{start}$. Conversely, the computational time required by the UGV is given by the difference between the timestamp at which the velocities are published to the ROS topic (t_{pub}) and the timestamp at which it receives the UDP frame (t_{rec}), i.e., $t_{comp_2 = t_{pub} - t_{rec}}$. These include the computational time needed by the waypoint generator to compute the appropriate (x, y, θ) coordinates and prepare the UDP frame for transmission (t_{comp_1}), the latency (t_{lat}), and the computational time required by the UGV to receive the UDP frame data and compute the appropriate linear and angular velocities (t_{comp_2}). All these factors influence the trajectory tracking process, as evidenced by the errors observed in x, y, and θ. The statistics of the errors are provided in TABLE I, representing the minimum achievable errors given that the computational times (t_{comp_1} and t_{comp_2}) are inherent and the latency t_{lat} is minimized, as this scenario has been tested in a localHost environment. Additionally, the time intervals values for each UDP frame t_{comp_1}, t_{lat}, t_{comp_2} and the overall delay are shown in Fig. 8.

In this situation, no UDP frame is considered as lost as all UDP frames are received prior to the scheduled time at which the UGV should reach the next waypoint.

	mean	std
$e_x[m]$	0.062	0.049
$e_y[m]$	0.079	0.061
$e_\theta[rad]$	0.222	0.763

TABLE I: Trajectory errors if no offset or latency are added.



(a) Computational time on the sender simulated device. (b) Latency between the sender and receiver simulated devices. (c) Computational time on the receiver simulated device. (d) Sum of t_{comp_1}, t_{lat} and t_{comp_2}.

Fig. 8: Time values for each UDP frame considering the case with no added offset and latency.

B. Second Scenario

In the second scenario, the tracked trajectory is depicted in Fig. 9.

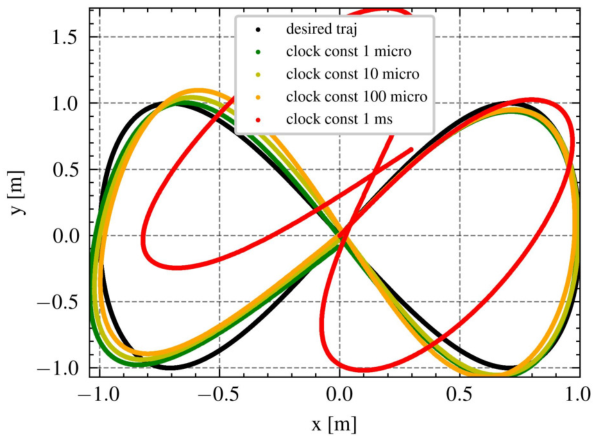


Fig. 9: Trajectory tracking results with constant offsets applied to the UGV clock.

Upon examination, it becomes evident that as the offset between the two clocks increases, the trajectory tracking performance deteriorates. This degradation is also reflected in the errors observed in the x, y, and θ coordinates corresponding to the varying time offsets, as detailed in TABLE II, III, IV, and V.

	mean	std
$e_x[m]$	0.058	0.044
$e_y[m]$	0.071	0.057
$e_\theta[rad]$	0.200	0.717

TABLE II: Trajectory errors when an offset of $1\mu s$ is added.

	mean	std
$e_x[m]$	0.061	0.046
$e_y[m]$	0.078	0.051
$e_\theta[rad]$	0.207	0.720

TABLE III: Trajectory errors when an offset of $10\mu s$ is added.

	mean	std
$e_x[m]$	0.074	0.047
$e_y[m]$	0.107	0.063
$e_\theta[rad]$	0.250	0.806

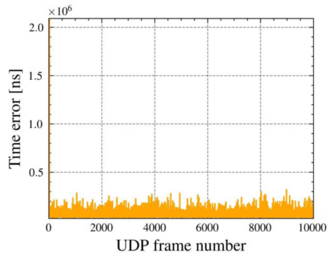
TABLE IV: Trajectory errors when an offset of $100\mu s$ is added.

	mean	std
$e_x[m]$	0.337	0.310
$e_y[m]$	0.391	0.347
$e_\theta[rad]$	0.530	0.872

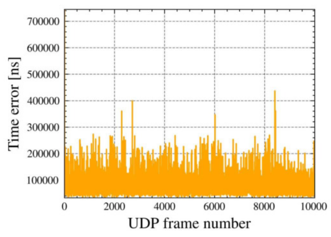
TABLE V: Trajectory errors when an offset of $1ms$ is added.

In these scenarios, synchronization errors between the clocks contribute to trajectory tracking inaccuracies, with the magnitude of the error directly proportional to the synchronization error value. Specifically, the comprehensive time encompassing the computation, creation, transmission, reception, and analysis of each UDP frame is illustrated in Fig. 10.

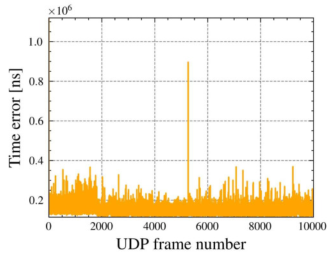
Notably, the overall time is displayed since the computational times on both the sender and receiver devices remain nearly constant across all four considered cases.



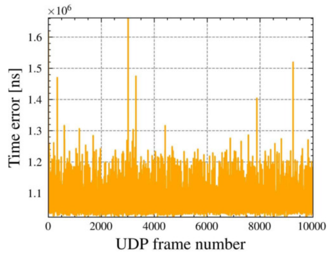
(a) Adding a constant offset equal to $1\mu s$.



(b) Adding a constant offset equal to $10\mu s$.



(c) Adding a constant offset equal to $100\mu s$.



(d) Adding a constant offset equal to $1ms$.

Fig. 10: Overall delay for each UDP frame considering the scenarios with constant added offsets.

As in the previous instance, no UDP frame is lost because the new UDP frame is received by the UGV within a maximum of 10 ms.

C. Third Scenario

In the third scenario, we simulate the offset accordingly with the real data acquired as explained in Section 3.C. Looking at the results in Fig. 11, we can see that the tracked trajectory almost coincides with the one given without taking into account any offset (Fig. 7) in agreement with the fact that the mean and standard deviation of the acquired data are of $-647\ ns$ and $2776\ ns$, respectively.

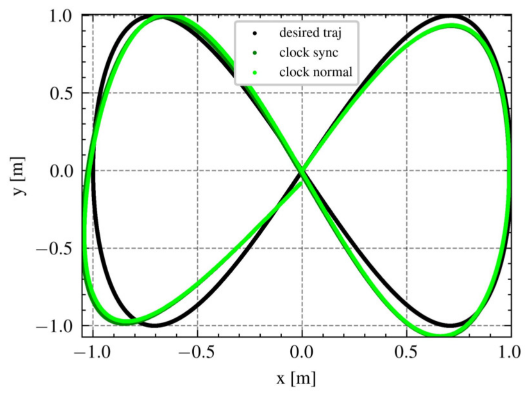


Fig. 11: Trajectory tracking result given by adding to the UGV clock the random values from the normal distribution.

Also the errors of the x, y, θ coordinates of the tracked trajectory are acceptable, as they are comparable with the ones given without any added offset or latency, as demonstrated by the mean and standard deviation of such errors, as shown in TABLE VI.

	mean	std
$e_x[m]$	0.071	0.49
$e_y[m]$	0.100	0.071
$e_\theta[rad]$	0.278	0.859

TABLE VI: Trajectory errors when an offset based on acquired real data is added.

Such error values are derived by the offset synchronization and the time required to compute, send, and analyze each UDP frame. The total time to do so for each UDP frame and the contribution of each component, i.e., t_{comp_1}, t_{lat} and t_{comp_2}, are given in Fig. 12.

Coherently with such results, no UDP frame is lost in the transmission.

D. Fourth Scenario

In the fourth scenario, the tracked trajectories are shown in Fig. 13.

Observing this, it becomes apparent that the tracked trajectory does not align with the desired trajectory in either case. This result is confirmed by the coordinate errors characterized by an increasing offset of $1\mu s$ and $10\mu s$ provided in TABLE VII and VIII, respectively.

Taking into account the case where we added $1\mu s$ for every received UDP frame, knowing that 10000 UDP frames are

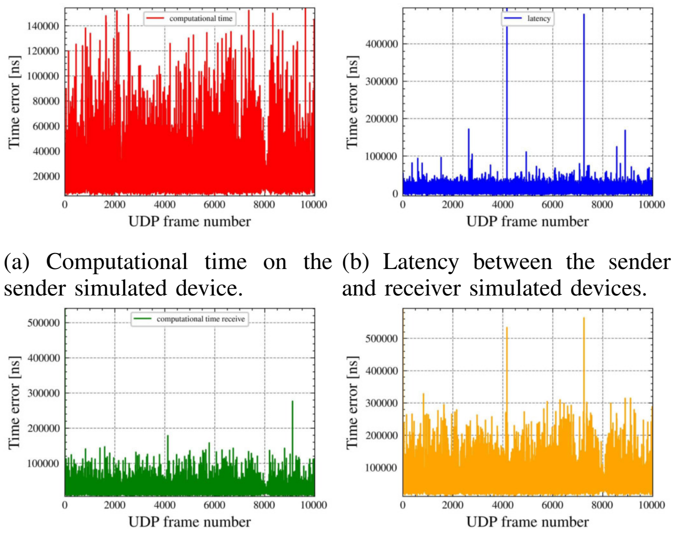

(a) Computational time on the sender simulated device.
(b) Latency between the sender and receiver simulated devices.
(c) Computational time on the receiver simulated device.
(d) Sum of t_{comp_1}, t_{lat} and t_{comp_2}.

Fig. 12: Overall delay for each UDP frame considering the scenario adding an experimental normal offset.

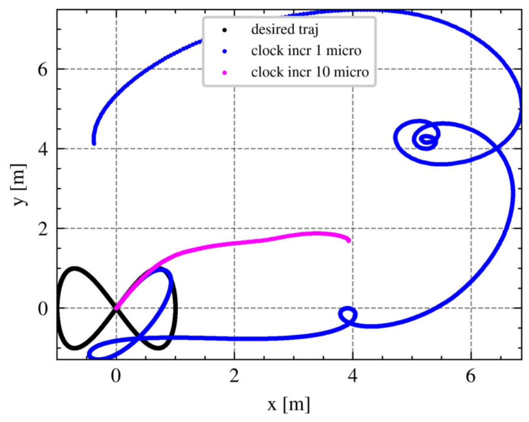


Fig. 13: Trajectory tracking results given by considering a direct proportionality between the UDP frame number and the offsets of $1\mu s$ and $10\mu s$, respectively.

sent, the UGV can move in the environment since the time instant at which the UDP frame is received is always before the last desired time at which the UDP frame has to be in the next waypoint. Instead, in the second scenario, verified when we added $10\mu s$ to the received time instant times the

	mean	std
$e_x[m]$	2.809	2.738
$e_y[m]$	1.639	2.184
$e_\theta[rad]$	1.435	1.238

TABLE VII: Trajectory errors when an offset of $1\mu s$ times the UDP frame number is added.

	mean	std
$e_x[m]$	3.546	1.363
$e_y[m]$	1.618	0.830
$e_\theta[rad]$	1.448	1.475

TABLE VIII: Trajectory errors when an offset of $10\mu s$ times the UDP frame number is added.

number of the UDP frame itself, it is verified that at a certain point the desired time instant received by the UGV results in the past with respect to the actual time instant at which the UDP frame is received. As a consequence, the UGV stops and cannot move further.

E. Fifth Scenario

The last experiments focus on introducing latency into the communication between the simulated devices to emulate a more realistic scenario, closely resembling what would be encountered under real-world conditions. The impact of latency on the trajectory tracking problem was explored using both the controllers to compute the suited linear and angular velocities, with the results depicted in Fig. 14.

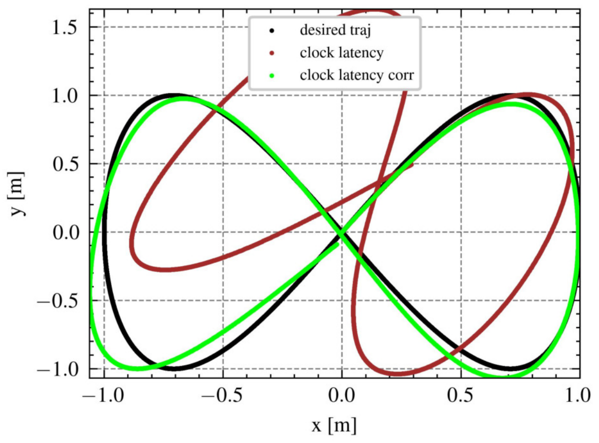


Fig. 14: Trajectory tracking results obtained adding latency given by acquired experimental data using the first (in brown) and the second controller (in lime).

As anticipated, incorporating latency into the system and adopting the first controller, i.e., assuming an unknown sending time instant t_{send}, leads to increased errors in the x, y, and θ coordinates of the UGV with respect to the case in which the sending timestamp is known as provided in TABLE IX.

	mean	std
$e_x[m]$	0.295	0.285
$e_y[m]$	0.359	0.318
$e_\theta[rad]$	0.492	0.784

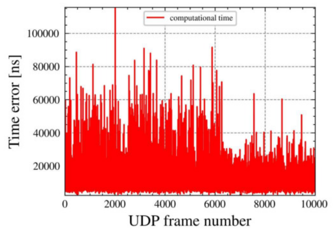
TABLE IX: Trajectory errors if latency is added and $Controller_1$ is employed.

However, by leveraging time synchronization standards and integrating knowledge of t_{send} while employing the second controller described in Section 3.D, the UGV is capable of tracking the desired trajectory, once again achieving performance levels equivalent to those obtained without any added offsets or latency in the system as shown in TABLE X.

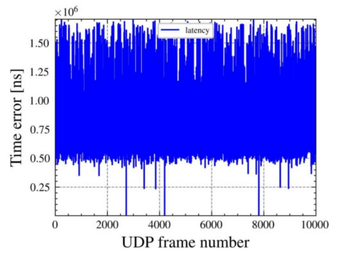
Even if, in both the two situations, the overall delays are the same and their values are given in Fig. 15, thanks to time synchronization we are able to manage the time and adjust the UGV clock fulfilling again the task requirements.

	mean	std
$e_x[m]$	0.051	0.047
$e_y[m]$	0.048	0.045
$e_\theta[rad]$	0.130	0.550

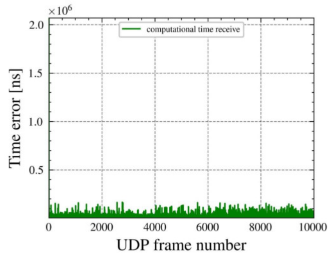
TABLE X: Trajectory errors if latency is added and $Controller_2$ is employed.



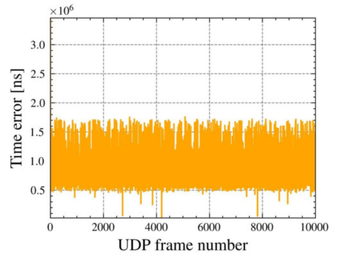
(a) Computational time on the sender simulated device.



(b) Latency between the sender and receiver simulated devices.



(c) Computational time on the receiver simulated device.



(d) Sum of t_{comp_1}, t_{lat} and t_{comp_2}.

Fig. 15: Overall delay for each UDP frame considering the scenario adding experimental acquired latency values.

V. Conclusions

In conclusion, our study underlines the central role of time synchronization in trajectory-tracking scenarios, particularly in contexts where temporal precision is paramount. Through the shown experiments, we highlight the impact of synchronization errors on system performance, affirming the criticality of precise clock alignment.

Moreover, we have enhanced the ability of the IEEE 802.1AS standard to mitigate synchronization impairments induced by latency and other factors. By leveraging this standard, we can effectively correct timing discrepancies, reaching again the required task requirements and reinstating system performance to the desired level.

Ultimately, the research demonstrates the intrinsic value of temporal awareness in navigating time-dependent variables within the system. In fact, by understanding and effectively managing time synchronization, we are able to demonstrate how the knowledge of time can allow to fulfil the task scope, despite the time-dependent factors that act on the system.

References

[1] S. Mumtaz, A. Al-Dulaimi, V. Frascolla, S. A. Hassan, and O. A. Dobre, "Guest Editorial Special Issue on 5G and Beyond—Mobile Technologies and Applications for IoT," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 203–206, 2019.

[2] S. Sudhakaran, C. Hall, D. Cavalcanti, A. Morato, C. Zunino, and F. Tramarin, "Measurement method for end-to-end Time synchronization of wired and wireless TSN," in *2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2023, pp. 1–6.

[3] L. M. Camarinha-Matos, R. Fornasiero, and H. Afsarmanesh, "Collaborative networks as a core enabler of industry 4.0," in *Collaboration in a Data-Rich World*, L. M. Camarinha-Matos, H. Afsarmanesh, and R. Fornasiero, Eds. Cham: Springer International Publishing, 2017, pp. 3–17.

[4] J. Hicking, M.-F. Stroh, and S. Kremer, "Collaboration through digital integration – an overview of it-ot-integration use-cases and requirements," in *Smart and Sustainable Collaborative Networks 4.0*, L. M. Camarinha-Matos, X. Boucher, and H. Afsarmanesh, Eds. Cham: Springer International Publishing, 2021, pp. 403–410.

[5] A. Larrañaga, M. C. Lucas-Estañ, I. Martinez, I. Val, and J. Gozalvez, "Analysis of 5g-tsn integration to support industry 4.0," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1111–1114.

[6] I. Val, Seijo, R. Torrego, and A. Astarloa, "Ieee 802.1as clock synchronization performance evaluation of an integrated wired–wireless tsn architecture," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 2986–2999, 2022.

[7] A. Merwaday, R. Vannithamby, M. Eisen, S. Sudhakaran, D. A. Cavalcanti, and V. Frascolla, "Communication-Control Co-design for Robotic Manipulation in 5G Industrial IoT," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6.

[8] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (tsn)," *IEEE Access*, vol. 11, pp. 61 192–61 233, 2023.

[9] Y. Kang, S. Lee, S. Gwak, T. Kim, and D. An, "Time-sensitive networking technologies for industrial automation in wireless communication systems," *Energies*, vol. 14, no. 15, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/15/4497

[10] C. J. Bernardos, A. Mourad, M. Groshev, L. M. Contreras, M. M. Roselló, O. Bularca, V. Frascolla, P. Szilagyi, and S. Robitzsch, "Using RAW as Control Plane for Wireless Deterministic Networks: Challenges Ahead," in *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MobiHoc '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 328–333. [Online]. Available: https://doi.org/10.1145/3565287.3617608

[11] S. Sudhakaran, I. Ali, M. Eisen, J. Perez-Ramirez, C. Cazan, V. Frascolla, and D. Cavalcanti, "Zero-delay roaming for mobile robots enabled by wireless tsn redundancy," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[12] V. Frascolla, D. Cavalcanti, and R. Shah, "Wi-Fi Evolution: The Path Towards Wi-Fi 7 and Its Impact on IIoT," *Journal of Mobile Multimedia*, 09 2022.

[13] H. Liao, Z. Zhou, Z. Yao, Z. Mumtaz, and V. Frascolla, "Information timeliness guaranteed communication and energy control integration in multi-mode power iot," 12 2023, pp. 2614–2619.

[14] J. Lee and S. Park, "Time-sensitive network (tsn) experiment in sensor-based integrated environment for autonomous driving," *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1111

[15] "Ntp: The network time protocol," Apr. 2024, http://www.ntp.org/.

[16] I. W. Paper, "Implementing real-time system using intel time-sensitive networking capable ethernet controller on linux operating system," https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2022-10/tsn-real-time-profinet-linux.pdf, 2022, accessed: 2024-04-10.

[17] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless time sensitive networking impact on an industrial collaborative robotic workcell," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7351–7360, 2022.

[18] S. Sudhakaran, V. Mageshkumar, A. Baxi, and D. Cavalcanti, "Enabling qos for collaborative robotics applications with wireless tsn," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.