ACTA IMEKO
ISSN: 2221-870X
March 2026, Volume 15, Number 1, 1 – 11

A time-sensitive networking-enabled measurement approach to latency assessment in real-time wireless networks

Alberto Morato[1], Elena Ferrari[2], Federico Tramarin[3], Stefano Vitturi[1]

[1] *Consiglio Nazionale delle Ricerche (CNR) – IEIIT, Padova, Italy*
[2] *University of Padova, Department of Information Engineering, Padova, Italy*
[3] *University of Modena and Reggio Emilia, Department of Engineering "Enzo Ferrari", Modena, Italy*

ABSTRACT
Networked measurements are essential for the design and implementation of cooperative cyber-physical systems in the Industry 4.0/5.0 era. Indeed, the need for accurate models of real-time communication systems and, especially, of their latency is paramount for enabling technologies such as advanced manufacturing solutions, simulation, and industrial internet. This paper presents a novel approach to measuring latency in wireless communication networks, focusing on the metrological characterization of individual subsystem contributions. By leveraging Time-Sensitive Networking (TSN) for synchronization, and employing cost-effective hardware solutions, we aim to provide a structured assessment of the measurement system's capability to isolate and quantify latency contributions of individual communication modules, from the protocol stack to the Wireless Network Interface Controller (WNIC). The proposed approach is practical, flexible, and adaptable to various environments and deployment scenarios, representing a significant advancement in latency measurement techniques for modern network environments.

Section: RESEARCH PAPER

Keywords: latency measurement; time-sensitive networking (TSN); networked measurement systems; wireless communications; digital twins; Industry 4.0/5.0

Citation: A. Morato, E. Ferrari, F. Tramarin, S. Vitturi, A time-sensitive networking-enabled measurement approach to latency assessment in real-time wireless networks, Acta IMEKO, vol. 15 (2026) no. 1, pp. 1–11. DOI: 10.21014/actaimeko.v15i1.2220

Section Editor: Francesco Lamonaca, University of Calabria, Italy

Received: October 3, 2025; **In Final Form:** November 7, 2025; **Published:** March, 2026.

Funding: This research has been supported in part within the activities of the project "Time Sensitive Networking for future enabling technologies: measurement methods and metrological characterization in hybrid wired/wireless scenarios" CUP E53D23014650001, funded by EU in NextGenerationEU plan through the Italian "Bando Prin 2022 PNRR - D.D. 1409 del 14-09-2022" by MUR.

This study was partially 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). This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Corresponding Author: Alberto Morato, e-mail: alberto.morato@cnr.it

1. INTRODUCTION

In the era of Industry 4.0/5.0, cyber-physical systems must collaboratively make decisions despite uncertainties arising from sensor data, network impairments, and actuation. Control decisions must remain robust across heterogeneous communication domains to support human-machine and machine-to-machine collaboration. Networked measurements are essential for the design and implementation of these systems, which require the availability of accurate theoretical and simulation models for real-time communication systems.

These models are crucial for the virtual emulation of physical systems, especially for digital twins, enabling system design, optimization, and monitoring without direct physical intervention [1], [2]. While latency estimation and modeling are well-established topics, they often remain an open issue from the modeling point of view due to the continuous evolution of network traffic, topologies, and application demands [3]. Precise latency distributions are paramount for building and tuning theoretical models effectively, as latency serves not only as a critical performance indicator but also as a means of detecting safety and security issues in sensor networks [4], [5].

Network latency modeling is generally characterized by a black-box approach [6], focusing solely on end-to-end latency [7], [8]. However, this approach often overlooks the complexities of the underlying communication infrastructure. To accurately assess latency in time-critical systems, it is essential to consider the specific and characteristic delay contributions introduced by each subsystem within the entire communication infrastructure [7], [9].

However, this approach presents several challenges. Firstly, not all subsystems are directly measurable, even with software or hardware probes. For instance, modern Network Interface Controllers (NICs) often utilize closed-source firmware and specialized hardware, rendering internal measurements impractical [10]. Secondly, time synchronization is crucial in this measurement process, particularly for end-to-end latency measurements and NIC latency characterization [11], [12]. These aspects become even more challenging in wireless networks, where capturing transmissions is inherently more complex.

In this paper, we present a novel approach to measure latency in wireless communication networks, with the ultimate aim of providing a metrological characterization of latency contributions from the protocol stack to the Wireless Network Interface Controller (WNIC). Our system is based on cost-effective hardware solutions and exploits a supplementary wireless device in monitor mode to capture transmitted packets. We leverage the IEEE 802.1AS standard within Time-Sensitive Networking (TSN) for time synchronization, enabling precise timestamp coordination across all devices involved in the measurement process. TSN provides low-latency and deterministic communication over Ethernet [13] and has recently been extended to the wireless domain [14].

The paper is organized as follows. Section 2. investigates the main related works and delineates the actual paper contributions. Section 3. describes the architecture of the measurement setup and the proposed measurement methodology. Section 4. provides a mathematical modeling of the measurement system. Section 5. applies the proposed technique to real-case scenarios and discusses the obtained results. Finally, section 6. provides some concluding remarks.

2. RELATED WORKS AND CONTRIBUTION

Latency measurement is a well-established field with numerous approaches and methodologies. Probe packets, specifically forged and timestamped to measure communication infrastructure delays, are a common instrument for latency evaluation [5]. For instance, [15] exploits lightweight probe packets at link edges, while [16] introduces DMS, a statistical latency measurement platform leveraging DNS infrastructure in large-scale scenarios. Paper [17] combines TSN and probe packets to detect issues in Software-Defined Networking applications. With a different approach, [18] explores the correlation between network- and application-level latency using packets directly involved in the final application.

Accounting for delays induced by the measurement system itself is crucial. To this aim, [19] proposes INTEST, a method to compensate for probe packet delays in wired networks, whereas [20] addresses packet delay estimation in distributed service networks, aiming to reduce overheads while maintaining accuracy. In [21], the authors introduce COLATE, a probe-less scheme that yields precise per-flow latency estimates through statistical denoising.

Timestamping probes at different points along the end-to-end communication path requires maintaining mutual synchronization among probes. Several studies have explored this method, often using GPS for synchronization purposes in diverse scenarios, from power grids to software systems, as described in [22]–[25]. Conversely, [26] evaluates the performance of IEEE 1588 for time synchronization in transmission substations, concluding that it is accurate and suitable for protection and control applications, potentially more so than GPS-only systems.

Despite these advancements, most existing approaches primarily focus on end-to-end latency, overlooking the contributions of individual subsystems.

2.1. Contribution

Moving from the above considerations, our research extends beyond the limitations of existing approaches by enabling the measurement of each subsystem's contribution to overall latency. The primary objective is to define a measurement methodology that captures latency introduced by both the protocol stack and the WNIC during transmission and reception phases, as well as accurately measure and account for the packet transmission process.

A secondary, yet equally important, goal is to provide a cost-effective solution. This is achieved by utilizing an independent wireless device in monitor mode and eliminating the need for GPS receivers on every device, a common limitation in existing approaches. Indeed, while GPS technology could offer the required sub-microsecond precision [27], [28], its practicality may be limited in several scenarios. Therefore, we leverage TSN and the IEEE 802.1AS standard for time synchronization, enabling precise synchronization across all network devices without GPS modules or hardware-level interventions.

3. SYSTEM ARCHITECTURE AND MEASUREMENT METHODOLOGY

The measurement setup architecture, illustrated in Figure 1, comprises four general-purpose devices. Two of these devices, designated as the source and destination, are directly linked via Wi-Fi and serve as the devices under test (DUTs). These DUTs are used to evaluate both end-to-end wireless latency and individual contributions of each subsystem involved in the communication process. The third component is a "capture device", i.e. an external WNIC, transparent to the main communication, and equipped with the industry-standard Wireshark packet analysis tool.

Since the expected latencies are in the order of hundreds of microseconds, it becomes important to maintain precise synchronization across the devices to enable accurate measurements. To address this, the fourth component of the setup is a TSN switch. This switch implements a secondary Ethernet network that interconnects all devices, and is used to ensure a strict synchronization through the standard IEEE 802.1AS, a generalized profile of the IEEE 1588 Precision Time Protocol (PTP) defined with TSN, and commonly referred to as "gPTP".

All devices in the setup operate on the Linux kernel, chosen for its comprehensive TSN support and ability to provide access to both hardware and software timestamping capabilities. These features are essential for the proposed measurement methodology.

3.1. Details on the measurement procedure

With reference to Figure 1, the direct connection through a Wi-Fi link is the principal path between the source and destination DUTs. Specifically, the source device works as a SoftAP (Wi-Fi Access Point implemented via software), while the destination device operates as a Station. Meanwhile, the capture device passively monitors the wireless channel, overhearing transmitted frames on the channel without connecting to the SoftAP.

In this setup, latency measurements are implemented with the transmission of a stream of User Datagram Protocol (UDP) packets between the source and destination devices. Each UDP frame was embedded with a unique sequential number and a transmission timestamp [17]. These packets were then subjected

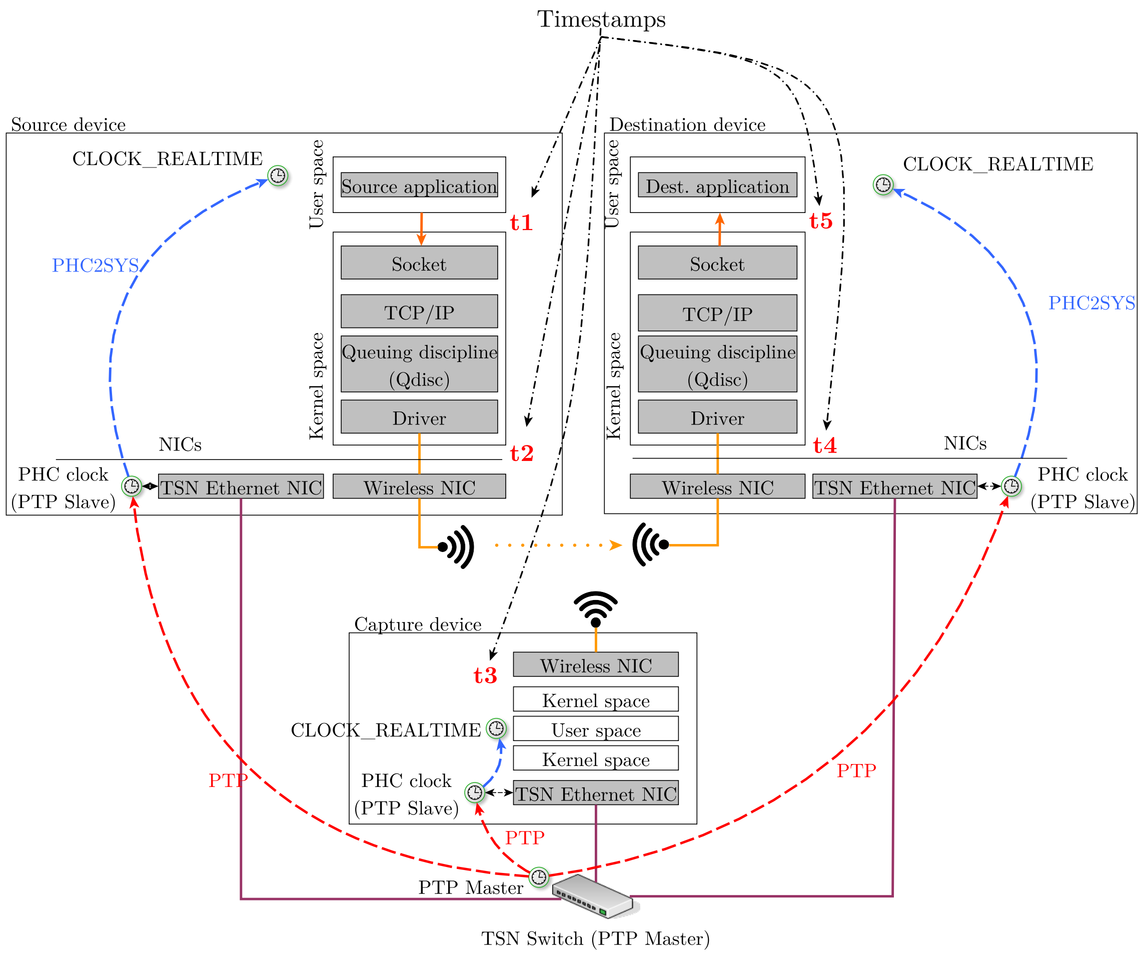


Figure 1. Scheme of the measurement setup.

to timestamping at multiple stages throughout the communication process, as it will be better detailed in subsection 3.2. This approach allows for a comprehensive analysis of latency at various points in the network transmission path, with the definition of three distinct datasets:

1) At receiver side: containing t_1 and t_5 timestamps (representing end-to-end latency).
2) At source side: containing t_2 timestamps.
3) At capture device: containing t_3 timestamps.

Once these datasets have been acquired, they are merged by comparing the sequence number of each packet. The obtained final dataset not only yields complete latency information but also provides insights into possible packet losses and the stage at which a packet was dropped or lost. In practice, a comprehensive view of the communication process was obtained by correlating timestamps and sequence numbers across the datasets.

3.2. Measurement probes

In the proposed measurement setup, several software probes have been deployed at different points along the communication path. These probes are referred to as t_X, where X ranges from 1 to 5, and serve to measure the intermediate times at various stages, as better detailed below.

Two probes, namely t_1 and t_5, are positioned at the boundary between the application level and the kernel space of the source and destination devices, respectively. At transmission, the packet is timestamped with t_1 just before being passed to the protocol stack, while at reception, the packet is timestamped at the application level as soon as it becomes available from the kernel space. For this reason, t_1 and t_5 do not include any additional (random) processing time due to application protocol execution, and they are retrieved by reading the `CLOCK_REALTIME`.

Another pair of probes, t_2 and t_4, are positioned at the bottom of the kernel space on both the source and destination devices. These timestamps are obtained using the `SO_TIMESTAMPING` socket option, which allows the Linux kernel to provide high-precision transmit and receive timestamps [29]. It is a Linux feature that enables programs to ask the system (kernel) for precise timing information about when a packet was actually transmitted or received. If the hardware supports this feature, which is the case for Network Interface Cards (NICs) equipped with PTP Hardware Clocks (PHCs), this mechanism captures timestamps very close to the physical layer and without OS intervention, significantly reducing the impact of processing delays on the timestamp value.

This mechanism allows precise timestamping at the lowest level of the protocol stack. Specifically, transmit timestamps t_2 are generated by the NIC hardware at the moment a frame is physically transmitted, while receive timestamps t_4 are recorded upon physical packet arrival.

The final probe, t_3, is located on the capture device, which intercepts transmitted frames using a wireless interface operating in monitor mode. This configuration enables the capture of raw wireless frames directly from the air, independent of higher-layer protocol processing. Timestamping occurs just before the captured frames are handed to the kernel from the WNIC.

Timestamp t_3 measures the delay introduced by the WNIC on the source device during frame transmission. Specifically, after being timestamped at t_2, the frame is passed to the WNIC for transmission. Timestamp t_3 captures the exact moment the frame is transmitted over the air. Since the interface operates in monitor mode and is not part of a typical socket-based communication flow, Wireshark is the only feasible way to obtain timestamps in this context. This tool records the precise moment a frame is passed from the WNIC to the kernel driver in kernel space, offering a high-resolution timestamp aligned with the other probes in the system.

All the described probes, illustrated also in Figure 1, are fetched from the `CLOCK_REALTIME` clock internal to each device. In Linux, `CLOCK_REALTIME` refers to the system-wide real-time clock, which represents the current real-world time (i.e. ”wall clock” time) inside the system [30]. This clock provides high resolution timestamps with nanosecond precision and can be adjusted by external synchronization mechanisms, such as NTP (Network Time Protocol) or PTP.

In our setup, the IEEE 802.1AS-defined gPTP distributes a common time reference to all participating devices, allowing each instance of `CLOCK_REALTIME` to reflect a unified global time base. In particular, for the synchronization, the TSN switch serves as the Grandmaster (GM), not only providing connectivity but also acting as the time reference for all other devices. The DUTs and the capture device function as Slaves (SL) in this configuration. PTP, which is implemented through the `linuxPTP` suite, synchronizes the internal PTP Hardware Clock (PHC) of each Ethernet Network Interface Controller (NIC). As the PHC is not directly accessible for timestamping, it is necessary to synchronize it with the system's real-time clock (`CLOCK_REALTIME`). This synchronization is achieved using the `phc2sys` clock management software, which periodically reads the PHC and adjusts the system clock accordingly. As a result, each device's system clock is synchronized with the GM clock with nanosecond-level precision, ensuring accurate and consistent timekeeping across the entire network.

4. MODELING OF THE MEASUREMENT SYSTEM

To ensure that the latency values extracted from the datasets mentioned in section 3. are both meaningful and trustworthy, it is necessary to analyze how synchronization accuracy and timestamp uncertainties affect the measurements. Since latency is computed from timestamp differences between distinct devices, understanding how uncertainties propagate through the system is essential.

This section presents an analytical evaluation of whether the proposed setup can yield accurate and stable latency measurements. The analysis builds on the synchronization model introduced in [31], which describes master-slave synchronization using PTP with servo controllers.

We extend this model to account for multiple slave devices, as required by our experimental configuration. This is critical because, as discussed in the previous section, latency computation relies on timestamp data collected across multiple devices. Thus, assessing the synchronization mechanism's ability to constrain uncertainty and maintain consistency in a multi-slave scenario is fundamental.

The extended model is introduced at first in a simplified form, considering only one master and two slaves, as depicted in Figure 2. These slaves serve as a representative case and do not correspond directly to specific slaves in the actual proposed setup. Indeed, the analysis performed on this simplified two-slave model can be generalized to any pair of slaves within the larger configuration. In

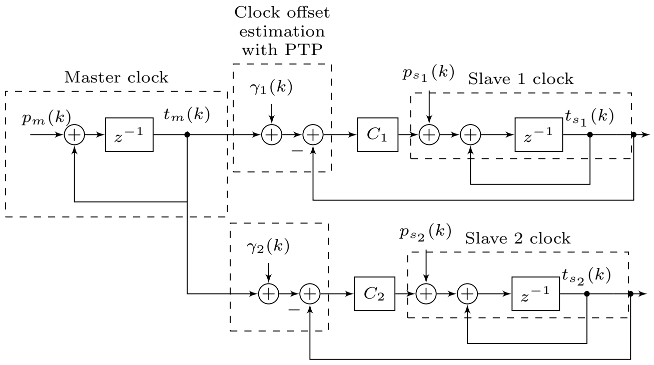


Figure 2. The proposed synchronization model (derived and extended from [31]). The model refers to the case of one master and to multiple slaves. Each slave is driven by a controller C_i which gets the master's offset estimated by PTP as an input.

the following, note that vector or sequence variables are presented in bold, whereas scalar variables are shown in regular font.

Following the original model, the master and slave clocks are modeled as discrete-time integrators, updated at each synchronization interval. The output sequences of the accumulators are denoted by $t_{\mathrm{m}}(k)$, for the master, and $t_{\mathrm{s_i}}(k)$, for the slave i, and represent their respective timestamps at step k. The corresponding input sequences to these integrators are denoted by $p_{\mathrm{m}}(k)$ and $p_{\mathrm{s_i}}(k)$, respectively, and are defined as follows:

$$\begin{aligned} p_{\mathrm{m}}(k) &= (1+\nu_{\mathrm{m}})\,\tau + \delta_\tau(k) + \varepsilon_{\mathrm{m}}(k) \\ p_{\mathrm{s_i}}(k) &= (1+\nu_{\mathrm{s_i}})\,\tau + \delta_\tau(k) + \varepsilon_{\mathrm{s_i}}(k) \end{aligned} \tag{1}$$

where:

a. τ: The nominal time period of the synchronization update;
b. ν_{m} and $\nu_{\mathrm{s_i}}$: The clock drift rates of the master and slave i, respectively, primarily due to local oscillator skew;
c. $\delta_\tau(k)$: The error in the synchronization period τ at step k, resulting from factors such as network delay variations (jitter) and congestion;
d. $\varepsilon_{\mathrm{m}}(k)$ and $\varepsilon_{\mathrm{s_i}}(k)$: Random sequences modeling the combined frequency and phase noise of the master and slave i clocks, respectively.

Comparing Figure 1 and Figure 2, it is important to note that $t_{\mathrm{m}}(\cdot)$ and $t_{\mathrm{s_i}}(\cdot)$ correspond to the PTP Hardware Clocks (PHCs) of the master and slave devices, respectively. It is then the responsibility of the `phc2sys` software to keep the system clock synchronized with the PHC inside each device. This will ensure that all timestamps are consistent across devices.

Each synchronization branch between the master and a slave device involves clock offset estimation using PTP. As in [31], we consider the PTP offset estimation at step k to be affected by a sequence of uncertainty contributions, denoted by $\gamma_i(k)$ for slave i. This term $\gamma_i(k)$ encapsulates various error sources:

- Propagation path asymmetries (recall that PTP fundamentally assumes symmetric forward and reverse communication paths);
- Timestamping jitter;
- Limited resolution of the timestamping process on both the master and slave devices.

The estimated offset then feed the controller C_i of the slave i, which is modeled as a proportional-integral (PI) controller.

From this, it is possible to derive the state-space realization of the system as in equation 2:

$$
\begin{gathered}
\text{Master} \\
\begin{cases} x_\mathrm{m}[k+1] = x_\mathrm{m}[k] + p_\mathrm{m}[k] \\ \qquad t_\mathrm{m}[k] = x_\mathrm{m}[k] \end{cases} \\[2ex]
\text{Slave 1} \\
\begin{cases} x_{\mathrm{s}_1}[k+1] = x_{\mathrm{s}_1}[k] + p_{\mathrm{s}_1}[k] + u_{\mathrm{s}_1}[k] \\ \qquad t_{\mathrm{s}_1}[k] = x_{\mathrm{s}_1}[k] \end{cases} \\[2ex]
\text{Slave 2} \\
\begin{cases} x_{\mathrm{s}_2}[k+1] = x_{\mathrm{s}_2}[k] + p_{\mathrm{s}_2}[k] + u_{\mathrm{s}_2}[k] \\ \qquad t_{\mathrm{s}_2}[k] = x_{\mathrm{s}_2}[k] \end{cases} \\[2ex]
\text{PI Controller 1} \\
\begin{cases} x_{\mathrm{c}_1}[k+1] = x_{\mathrm{c}_1}[k] + t_\mathrm{m}[k] - t_{\mathrm{s}_1}[k] + \gamma_1[k] \\ y_{\mathrm{c}_1}[k] = K_\mathrm{I}\, x_{\mathrm{c}_1}[k] + (K_\mathrm{P} + K_\mathrm{I})\,(t_\mathrm{m}[k] - t_{\mathrm{s}_1}[k] + \gamma_1[k]) \end{cases} \\[2ex]
\text{PI Controller 2} \\
\begin{cases} x_{\mathrm{c}_2}[k+1] = x_{\mathrm{c}_2}[k] + t_\mathrm{m}[k] - t_{\mathrm{s}_2}[k] + \gamma_2[k] \\ y_{\mathrm{c}_2}[k] = K_\mathrm{I}\, x_{\mathrm{c}_2}[k] + (K_\mathrm{P} + K_\mathrm{I})\,(t_\mathrm{m}[k] - t_{\mathrm{s}_2}[k] + \gamma_2[k]), \end{cases}
\end{gathered}
\tag{2}
$$

where the symbol $u_{\mathrm{s}_\mathrm{i}}$ represents the control input for the clock of the i-th slave. The internal clock states of the master and the i-th slave are denoted by x_m and $x_{\mathrm{s}_\mathrm{i}}$, respectively. Additionally, $x_{\mathrm{c}_\mathrm{i}}$ maintains the integral state of the PI controllers for each i-th slave.

It is noteworthy that all the PI controllers have identical gains K_P and K_I. This decision facilitates the analysis and constitutes a plausible assumption, particularly when the network devices employ an identical implementation of PTP. This scenario is applicable in our study, wherein we have utilized the `linuxPTP` implementation of PTP within our setup. Consequently, presuming homogeneous controller gains for both slaves is justifiable, given that in our implementation, the default configuration parameters for PTP have been retained and the same algorithm is deployed across all devices.

From equation 2, defining the state variable as $\mathbf{q} = \left[x_\mathrm{m}, x_{\mathrm{s}_1}, x_{\mathrm{s}_2}, x_{\mathrm{c}_1}, x_{\mathrm{c}_2}\right]^\mathrm{T}$ it is possible to derive the close-loop system dynamic that is:

$$
\begin{aligned}
\mathbf{q}^+ = {} & A\,\mathbf{q} + B_{\mathrm{p}_{\mathrm{s}_1}}\, p_{\mathrm{s}_1} + B_{\mathrm{p}_{\mathrm{s}_2}}\, p_{\mathrm{s}_2} + B_{\mathrm{p}_\mathrm{m}}\, p_\mathrm{m} \\
& + B_{_1}\,\gamma_1 + B_{_2}\,\gamma_2
\end{aligned}
\tag{3}
$$

with:

$$
\begin{aligned}
A &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ K_\mathrm{I} + K_\mathrm{P} & 1 - K_\mathrm{I} - K_\mathrm{P} & 0 & K_\mathrm{I} & 0 \\ K_\mathrm{I} + K_\mathrm{P} & 0 & 1 - K_\mathrm{I} - K_\mathrm{P} & 0 & K_\mathrm{I} \\ 1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 1 \end{bmatrix} \\
B_{\mathrm{p}_{\mathrm{s}_1}} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\mathrm{T} \\
B_{\mathrm{p}_{\mathrm{s}_2}} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^\mathrm{T} \\
B_{\mathrm{p}_\mathrm{m}} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^\mathrm{T} \\
B_{_1} &= \begin{bmatrix} 0 & K_\mathrm{I} + K_\mathrm{P} & 0 & 1 & 0 \end{bmatrix}^\mathrm{T} \\
B_{_2} &= \begin{bmatrix} 0 & 0 & K_\mathrm{I} + K_\mathrm{P} & 0 & 1 \end{bmatrix}^\mathrm{T}.
\end{aligned}
\tag{4}
$$

These equations form the basis for the determination of two key aspects of our system. First, we need to verify whether the system can minimize the difference between the master and slave clocks, i.e., x_m and $x_{\mathrm{s}_\mathrm{i}}$. Second, we need to assess the difference between the two slave clocks, x_{s_1} and x_{s_2}, as it is essential that they remain bounded. This is crucial because, as mentioned earlier, latencies are computed from the differences between timestamps recorded by different devices.

Regarding the first aspect, we refer to [31] where it is shown that if K_I and K_P are chosen such that the system remains stable, then:

$$
x_{\mathrm{s}_\mathrm{i}}(+\infty) - x_\mathrm{m}(+\infty) = 0 \quad \Rightarrow \quad x_{\mathrm{s}_\mathrm{i}}(+\infty) = x_\mathrm{m}(+\infty)\,. \tag{5}
$$

This implies that the slave clocks asymptotically converge to the master clock. Furthermore, it is shown that the steady-state covariance σ^2_{ms} of the difference between the master and slave clocks is given by:

$$
\sigma^2_{\mathrm{ms}_\mathrm{i}} = \frac{(2\,K_\mathrm{P}^2 + K_\mathrm{P}\,K_\mathrm{I} + 2\,K_\mathrm{I})\,\sigma^2 + 2\,(\sigma^2_{_\mathrm{m}} + \sigma^2_{_{\mathrm{s}_\mathrm{i}}})}{K_\mathrm{P}\,(4 - K_\mathrm{I} - K_\mathrm{P})}\,. \tag{6}
$$

Regarding the difference between two slave clocks, it may be useful to introduce a reduced state-space representation. Let us define a new set of state variables:

$$
\begin{aligned}
&\bar{\mathbf{q}} = \left[x_{\mathrm{s}_1} - x_{\mathrm{s}_2}, (x_{\mathrm{c}_1} - x_{\mathrm{c}_1}) + (\nu_{\mathrm{s}_1} - \nu_{\mathrm{s}_2})\,\tau/K_\mathrm{I}, x_\mathrm{m}\right]^\mathrm{T}, \\
&\bar{\mathbf{p}}_{\mathrm{s}_\mathrm{i}} = p_{\mathrm{s}_\mathrm{i}} - \varepsilon_{\mathrm{s}_\mathrm{i}}\,.
\end{aligned}
\tag{7}
$$

From the system equations in (2), the new state-space representation is given by:

$$
\begin{aligned}
\bar{\mathbf{q}}^+ = {} & \begin{bmatrix} 1 - K_\mathrm{P} - K_\mathrm{I} & K_\mathrm{I} & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{\mathbf{q}} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \varepsilon_{\mathrm{s}_1} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \varepsilon_{\mathrm{s}_2} \\
& + \begin{bmatrix} K_\mathrm{P} + K_\mathrm{I} \\ 1 \\ 0 \end{bmatrix} \gamma_1 + \begin{bmatrix} K_\mathrm{P} + K_\mathrm{I} \\ 1 \\ 0 \end{bmatrix} \gamma_2 \\
= {} & \bar{A}\bar{\mathbf{q}} + B_{\varepsilon_{\mathrm{s}_1}} \varepsilon_{\mathrm{s}_1} + B_{\varepsilon_{\mathrm{s}_2}} \varepsilon_{\mathrm{s}_2} + \bar{B}_{_1}\gamma_1 + \bar{B}_{_2}\gamma_2\,.
\end{aligned}
\tag{8}
$$

A first result from (8) is that the difference between two slave clocks does not depend on the master clock. In fact, all the components related to x_m inside the various matrices are equal to zero, meaning that it is possible to define a further reduced state-space representation that does not include x_m. Besides, the most significant result is that the difference between the slave clocks does not depend on the master clock.

By selecting the controller gains so that the eigenvalues λ_j satisfy $|\lambda_j| < 1$, we achieve zero steady-state tracking error between the slave clocks, i.e.,

$$
\begin{aligned}
\bar{q}_1(+\infty) = 0 \quad &\Rightarrow \quad x_{\mathrm{s}_1}(+\infty) - x_{\mathrm{s}_2}(+\infty) = 0 \\
&\Rightarrow \quad x_{\mathrm{s}_1}(+\infty) = x_{\mathrm{s}_2}(+\infty).
\end{aligned}
\tag{9}
$$

Now, considering the covariance matrix $P(k) = E\left\{\mathbf{q}(k)\,\mathbf{q}(k)^\mathrm{T}\right\}$ we have

$$
\begin{aligned}
P(k+1) = {} & A\,P(k)\,A^\mathrm{T} + B_{_{\mathrm{s}1}} B^\mathrm{T}_{\varepsilon_{\mathrm{s}1}} \sigma^2_{\varepsilon_{\mathrm{s}1}} + B_{_{\mathrm{s}2}} B^\mathrm{T}_{_{\mathrm{s}2}} \sigma^2_{_{\mathrm{s}2}} \\
& + \bar{B}_{\gamma_1} \bar{B}^\mathrm{T}_{\gamma_1} \sigma^2_{\gamma_1} + \bar{B}_{\gamma_2} \bar{B}^\mathrm{T}_{\gamma_2} \sigma^2_{\gamma_2}.
\end{aligned}
\tag{10}
$$

Under the assumption that the system is stable, there exist $\bar{k} | P(k+1) = P(k), \forall k > \bar{k}$ so that is possible to define the

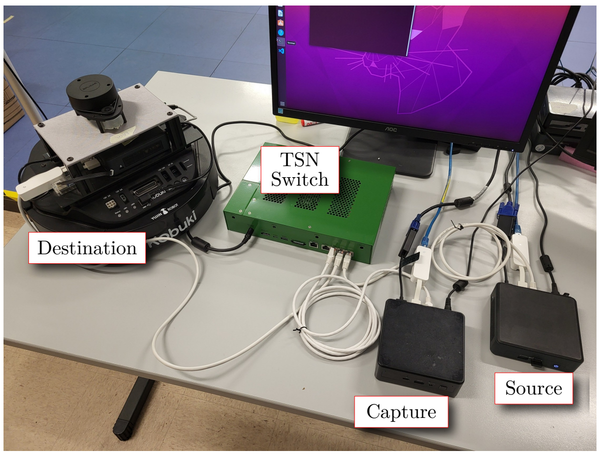


Figure 3. Real experimental setup.

steady-state covariance matrix $P(+\infty)$ in which the steady-state covariance $\sigma_{q_1}^2$ is the element $(1,1)$ of the matrix $P(+\infty)$:

$$\sigma_{q_1}^2 = \frac{\left(2\,K_P^2 + K_I\,K_P + 2\,K_I\right)\left(\sigma_{\gamma_1}^2 + \sigma_{\gamma_2}^2\right) + 2\left(\sigma_{s_1}^2 + \sigma_{s_2}^2\right)}{K_P\;(4 - K_I - 2\,K_P)}. \tag{11}$$

As a first observation, please notice that (6) and (11) share the same structure. Additionally, it is clear in equation (11) that the steady-state covariance of the difference between two slave clocks does not depend on the master clock in any term. Instead, it solely depends on their stability, i.e., the frequency and phase noise of the slave clocks, as well as the PTP offset estimation error on the respective slaves.

Similarly to what has been obtained in [32], given that $K_P > 0$, $K_I \geq 0$ and $\sigma_{(\cdot)}^2 > 0$, it follows that $0 < K_P < 2$ and $K_I < 4 - 2\,K_P$. So, satisfying these conditions, as soon as the uncertainties associated with the single slaves are bounded, the system is stable and the steady-state covariance of the difference between the two slave clocks is bounded.

As stated above, the slaves share the same implementation of PTP and, for the experimental sessions, the controller parameters were left at the default values. In particular, the default values for the `linuxPTP` implementation are $K_P = 0.7$ and $K_I = 0.3$. Clearly, the default gains satisfy the above conditions.

For the various devices involved in the setup, we do not have exact values of the associated uncertainties. Anyway, from [31]–[35], for non GPS-disciplined clocks, as in this measurement setup, we can assume $\gamma_i(k) \sim \mathcal{N}(0, \sigma_{\gamma_i}^2)$ and $\varepsilon_{s_i}(k) = \varepsilon_m(k) = \varepsilon(k) \sim \mathcal{N}(0, \sigma^2)$ with $\sigma_{\gamma_i} = 30$ ns and $\sigma^2 = 2$ ns.

Using these values and equations (6) and (11), we can compute the steady-state covariance of the difference between the two slave clocks and between the master and slave clocks which results respectively in $\sigma_{q_1} \approx 45$ ns and $\sigma_{ms} \approx 32$ ns.

5. EXPERIMENTAL RESULTS

The proposed measurement setup has been tested in a real–use case scenario, which is shown in Figure 3.

Specifically, both the source and destination devices are TNKi5000 Intel NUCs equipped with Intel Core i5-1135G7 CPUs, running Ubuntu 20.04 with kernel version 5.17. The source device was set as a SoftAP employing the Hostapd software. The capture device is a third Intel NUC but running Ubuntu 22.04 with kernel version 6.3. To set the capture device in monitor mode, we used the Airmon-ng utility provided by the Aircrack-ng suite.

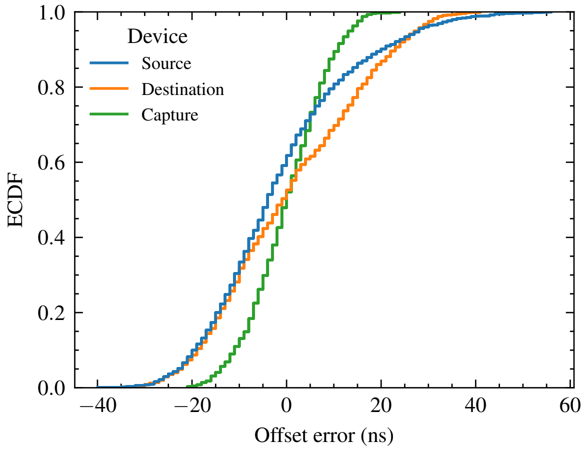


Figure 4. ECDF of the synchronization error.

Table 1. Statistics of the synchronization error.

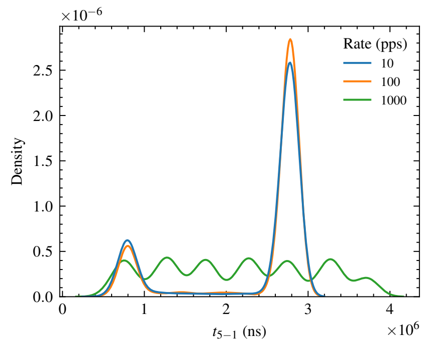
Device	Mean (ns)	Std (ns)	Min (ns)	Max (ns)
Capture	0.03	8.18	-21.00	24.00
Destination	0.72	15.69	-32.00	41.00
Source	-1.67	15.34	-40.00	56.00

The TSN switch is an NXP LS1028A-RDB, running the Real Time Edge Linux distribution. As mentioned, the switch is configured as the grandmaster clock, while the Intel NUCs are configured as slaves.

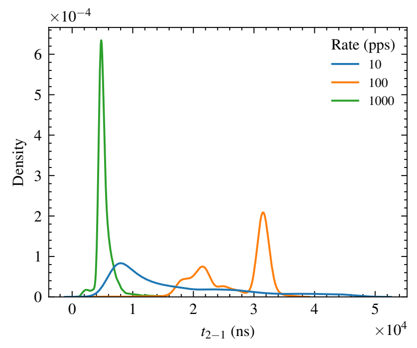
The first experiments concerned time synchronization since it is necessary to check the accuracy provided by IEEE 802.1AS. In this direction, we acquired the offset error reported by PTP for each slave. The results are shown in Figure 4 whereas the detailed statistics are reported in Table 1.

As can be seen, the time synchronization component operates well within acceptable limits, ensuring that data collection and analysis are accurate and reliable. Regardless of the device under consideration, the worst-case synchronization error consistently remains below 60 ns, with an average value centered around 0 ns. Notably, the standard deviation exhibits a non-negligible value on the order of tens of nanoseconds. This is due to the fact that the synchronization error is not constant but varies over time. This is a common behavior in PTP networks and is caused by several factors, including network conditions such as asymmetrical propagation delays and uncertainties in timestamping, the quality of the clock such as differences in skew, drift, jitter, and offsets, and the PTP update period, which in this case is set to 1 s. It is essential to note that the standard deviation found experimentally for each pair of master-slave devices refers to the standard deviation σ_{ms} estimated in (6). Notably, the experimental value is significantly lower than the theoretical one, indicating that the estimated σ_{ms} represents a worst-case scenario. This suggests that the uncertainty contributions from $\gamma_i(k)$, $\varepsilon_{s_i}(k)$, and $\varepsilon_m(k)$ in the real setup are considerably lower than those assumed in the model.

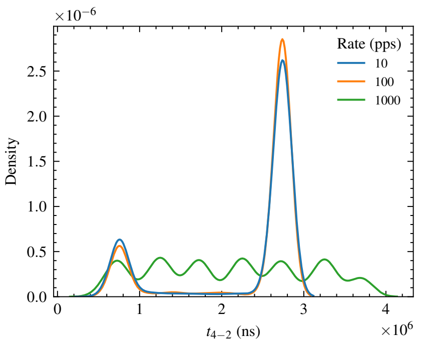
This is a positive result, as it confirms that the synchronization procedure effectively stabilizes the system and keeps uncertainties bounded. It is reasonable to assume that a similar behavior would be observed for slave-slave synchronization, even though it has not been verified experimentally. Such verification would require a more complex setup, which is beyond the scope of this work.



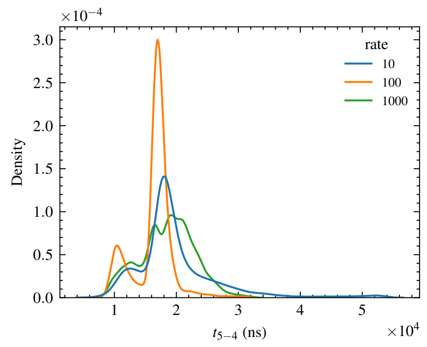
(a) EPDF of the end-to-end latency.



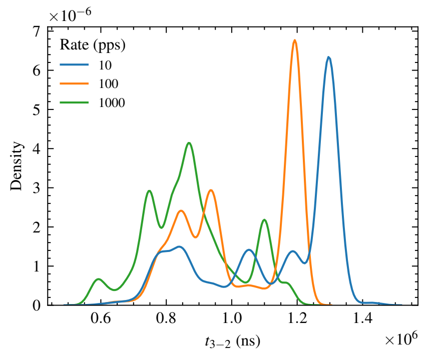
(b) EPDF of the transmission stack execution time.



(c) EPDF of the WNIC latency in TX and RX and the transmission medium delay.



(d) EPDF of the reception stack execution time.



(e) EPDF of the WNIC latency during the transmission.

Figure 5. EPDFs of different latencies in the system under test.

5.1. Multicast UDP packet stream

The second experiment was concerned with latency measurements. In particular, three sets of experiments were carried out, where probe frames were sent as a multicast UDP stream with different rates, specifically 1000 packets per second (pps), 100 pps, and 10 pps. We selected those rates as they cover the majority and most common use cases in the field of wireless industrial applications [36]. Using a multicast stream represents the most general way to exploit the proposed measurement setup. Indeed, this approach allows for the capture of the same probe frame by multiple devices. This is particularly useful for comparing latency when multiple destination devices or communication paths are involved. It allows for a direct comparison of timestamps captured by the probes, providing a comprehensive view of the transmission process. The focus was on

- t_{5-1} that accounts for the end-to-end latency;
- t_{2-1} that provides an insight about the latency introduced by the protocol stack during the transmission;
- t_{4-2} that both latencies of the WNIC in tx and rx, the transmission medium delay;
- t_{3-2} which provides an insight about the latency introduced by the WNIC during the transmission;
- t_{5-4} that accounts for the reception stack execution time.

The results are shown in Figures 5a to 5e whereas the detailed statistics are reported in Table 2.

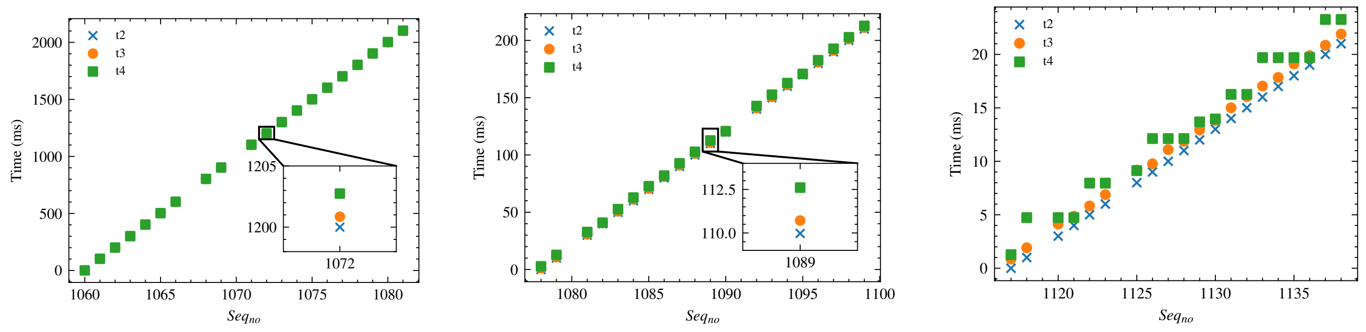
Examining the Empirical Probability Density Function (EPDF) end-to-end latency depicted in Figure 5a, it is evident that the latency trends can vary significantly depending on the packet rate. Specifically, at both 10 pps and 100 pps, we observe similar bimodal trends with very similar average values of 2344 µs and 2399 µs, respectively. However, this trend dramatically changes at 1000 pps, where the latency distribution becomes more dispersed, displaying an undulatory pattern while maintaining an average latency similar to the previous cases, albeit slightly lower. Analyzing the transmission stack execution time, shown in Figure 5b, and the value of t_{2-1} in Table 2, it can be observed that changing the packet rate leads to a significant change in latency. Specifically, as the packet rate increases, the transmission stack latency decreases. This may be due to power-saving mechanisms being temporarily disabled and/or the kernel prioritizing the sending process when high transmission rates are required. However, notably, a comparison with Figure 5a puts evidence that the contribution of the protocol stack execution time to the overall latency is minimal.

Table 2. Statistics of the latency measurements.

	Mean (µs)					
	t_{5-1}	t_{2-1}	t_{4-2}	t_{5-4}	t_{3-2}	t_{4-3}
rate (pps)						
10	2344	17	2308	20	1126	1182
100	2399	27	2356	16	1023	1333
1000	2126	5	2102	19	875	1227

Moving to Figure 5c, one of the most significant findings emerges from the analysis of WNIC latency in transmission and reception, along with the transmission medium delay.

Observing the trends, they closely mirror those of end-to-end latency, where the 10 pps and 100 pps rates exhibit a bimodal trend, while the 1000 pps rate displays a spread-out undulatory behavior. This measurement highlights the substantial influence of WNIC latency on end-to-end latency. It should be noted that Figure 5a



(a) 10pps (b) 100pps (c) 1000pps

Figure 6. Comparison packets timestamped in the source device kernel space (t_2), in the capture device (t_3) and in the destination device kernel space (t_4) at different packet rates.

and Figure 5c differ only by tens of microseconds, which roughly represents the delay introduced by the protocol stack. Leveraging the capture device allows for comprehensive insight into WNIC latency contributions in both transmission and reception.

Notably, the trends in Figure 5c and Figure 5e differ fundamentally, indicating that the trend in Figure 5c is not primarily driven by WNIC transmission latency. Further analysis revealed that latency t_{3-2} measured from the WNIC to the capture device is consistently lower than latency t_{4-2} measured from the WNIC to the destination device. This suggests that the contribution of the WNIC in transmission is generally slightly lower than in reception, indicating that the latency EPDF from WNIC to WNIC is mostly driven by the receiving WNIC, possibly due to queuing mechanisms. In other words, the WNIC in reception introduces a higher latency than the WNIC in transmission. It is the primary source for the bimodal behavior in the overall latency EPDF.

Overall, these results confirm that placing the WNIC of the capturing device in monitor mode significantly reduces latency while capturing raw packets, enabling effective measurement of WNIC latency in both reception and transmission.

Finally, the analysis of reception stack execution time depicted in Figure 5d reveals a latency similar to that of the transmission stack. However, unlike the transmission stack, changes in the transmission rate do not lead to significant alterations in the EPDF pattern.

In order to provide a more comprehensive understanding of latency dynamics, we compared packet timestamps at different stages of the communication process. Specifically, we compared timestamps at the source device kernel space (t_2), the capture device (t_3), and the destination device kernel space (t_4). The results, depicted in Figure 6, show the behavior across three different packet rates.

For 10 pps and 100 pps, the packets are sequentially transmitted, as evidenced by the timestamps acquired by the capture device occurring earlier than those at the destination device. This observation underscores the effectiveness of the proposed method with the capture WNIC in monitor mode.

However, the situation differs at the 1000 pps rate (Figure 6c). Here, it is evident that packets at the destination WNIC are enqueued before being passed to the kernel. Indeed, as can be seen, multiple packets with different sequence numbers are timestamped simultaneously. Interestingly, this effect is not observed in the timestamps of the source device, nor in those of the capture device. This is a clear indication that packet queuing is occurring at the destination WNIC.

Table 3. Statistics of the latency measurements with unicast UDP packet stream

	Mean (µs)					
	t_{5-1}	t_{2-1}	t_{4-2}	t_{5-4}	t_{3-2}	t_{4-3}
rate (pps)						
10	719	43	660	16	518	142
100	745	42	688	15	505	183
1000	686	7	664	15	436	228

5.2. Unicast UDP packet stream

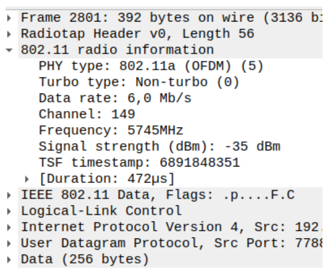
Another set of experiments was carried out using a unicast UDP packet stream. This has been done to investigate the sensitivity of the measurement setup to different mechanisms of generating probe frames. The same set of probes as in the previous section has been used to measure the same latencies.

The results are shown in Figure 8a, Figure 8b, Figure 8c, Figure 8d, and Figure 8e respectively, whereas, the detailed statistics are reported in Table 3.

Starting with the end-to-end latency shown in Figure 8a, it can be observed that it differs significantly from the multicast case shown in Figure 5a, both in terms of trends and mean latency. Regarding the trend, the multimodal behavior is still present. However, in this case, the EPDF for 1000 pps has the same trend as the other two rates. In other words, when unicast transmission is used, the oscillatory behavior at 1000 pps is no longer present, and the (bimodal) EPDFs for all three cases almost overlap. This means that, depending on the routing policies or the destination device, the network exhibits different behaviors.

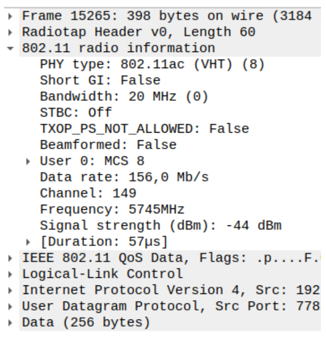
Regarding the mean latency, as seen by comparing Table 2 and Table 3, it is possible to observe a significant reduction in overall end-to-end latency in the unicast case, dropping from about 2300 µs for the multicast stream to about 700 µs for the unicast case. This result is not surprising. Indeed, in IEEE 802.11, multicast transmission is performed at a legacy data rate of 6 Mbps. In contrast, unicast transmission is either performed at the maximum data rate supported by the devices or selected by the rate adaptation algorithm. This is confirmed by the screenshot of packet acquisition reported in Figure 7, which compares the two raw captured frames by the capture device. As can be seen, in the multicast case, the transmission is performed at 6 Mbps, while in the unicast case, a rate of 156 Mbps is adopted.

Similarly to the multicast case, the transmission stack execution time, shown in Figure 8b, exhibits a decreasing trend as the packet rate increases. However, in this case, for 100 pps and 10 pps, the difference between the two EPDFs is not as significant as in the multicast case. On the other hand, similar to the multicast case, the



```
▸ Frame 2801: 392 bytes on wire (3136 bi
▸ Radiotap Header v0, Length 56
▾ 802.11 radio information
    PHY type: 802.11a (OFDM) (5)
    Turbo type: Non-turbo (0)
    Data rate: 6,0 Mb/s
    Channel: 149
    Frequency: 5745MHz
    Signal strength (dBm): -35 dBm
    TSF timestamp: 6891848351
  ▸ [Duration: 472μs]
▸ IEEE 802.11 Data, Flags: .p....F.C
▸ Logical-Link Control
▸ Internet Protocol Version 4, Src: 192.
▸ User Datagram Protocol, Src Port: 7788
▸ Data (256 bytes)
```

(a) Multicast



```
▸ Frame 15265: 398 bytes on wire (3184
▸ Radiotap Header v0, Length 60
▾ 802.11 radio information
    PHY type: 802.11ac (VHT) (8)
    Short GI: False
    Bandwidth: 20 MHz (0)
    STBC: Off
    TXOP_PS_NOT_ALLOWED: False
    Beamformed: False
  ▸ User 0: MCS 8
    Data rate: 156,0 Mb/s
    Channel: 149
    Frequency: 5745MHz
    Signal strength (dBm): -44 dBm
  ▸ [Duration: 57μs]
▸ IEEE 802.11 QoS Data, Flags: .p....F.
▸ Logical-Link Control
▸ Internet Protocol Version 4, Src: 192
▸ User Datagram Protocol, Src Port: 778
▸ Data (256 bytes)
```

(b) Unicast

Figure 7. A comparison raw captured frames by the capture device at different transmission types.

EPDF at 1000 pps shows a noticeable decrease in the transmission stack execution time.

It has to be stressed that in all experiments, all known and configurable power-saving settings were adjusted to limit the intervention of power-saving mechanisms. However, as suggested by Figure 8b, there appears to be a certain threshold in the packet transmission rate, beyond which the kernel allocates more resources to the transmission task, leading to the observed decrease in latency.

Regarding the reception stack execution time, shown in Figure 8d, the behavior is similar to the multicast case, suggesting that the reception stack execution time is not significantly affected by the packet rate, nor does it represent a significant contribution to the overall end-to-end latency.

As previously observed and confirmed by Figure 8c, the main contributor to the overall latency is the elaboration time of the WNICs. Indeed, analyzing Table 3, and considering the experiment at 10 pps, it can be noticed that 660 µs out of 719 µs are due to the WNICs. However, when examining each individual contribution to the WNIC latency t_{4-2}, namely t_{3-2} for the transmission and t_{4-3} for the reception, some fundamental differences with the multicast case are evident. Specifically, in the multicast case, t_{3-2} and t_{4-3} contribute approximately equally to t_{4-2}, while in the unicast case, the transmission latency t_{3-2} is noticeably greater than the reception latency t_{4-3}.

Overall, the obtained results allow us to draw some conclusions about the proposed methodology. Actually, the methodology has proven to be effective in characterizing the latency of individual subsystems within a communication network. It has demonstrated the ability to capture the contributions to the overall latency and to highlight differences in network behavior depending on the transmission type. In practice, the results show the flexibility of the proposed methodology, allowing it to capture details under various network conditions and application scenarios.

Notably, the results concerning the various measured latencies in the setup, particularly the one related to the airtime latency t_{4-2}, must be interpreted as best-case scenario values, as they are obtained in a quasi-ideal environment. Specifically, the distance between the source and destination devices is very short, and the Wi-Fi channel is selected to be free from interference or the presence of other transmitting devices. In a real-world scenario, airtime latency can vary significantly due to multiple factors, including the presence of other devices, increased distance between transmitter and receiver, obstacles in the environment, and the specific configuration of the Wi-Fi network and MAC layer. Since Wi-Fi relies on CSMA/CA, collisions may occur, forcing transmitting nodes to wait for a random backoff period before retransmitting. This introduces a non-deterministic component to the latency. Moreover, environmental conditions and MAC/PHY layer configurations directly influence the modulation scheme and data rate used for transmission. These parameters are also non-deterministic, as rate adaptation algorithms dynamically adjust the modulation scheme based on channel conditions. In general, longer distances or the presence of obstacles lead rate adaptation algorithms to select more robust but slower modulation schemes, thereby increasing airtime latency [37], [38]. Therefore, in more complex deployments, it is essential to consider the impact of the environment and network configuration when evaluating airtime latency [39], [40]. Nevertheless, the proposed methodology is capable of capturing airtime latency in real-world scenarios and can thus be employed to measure it in more complex setups.

6. CONCLUSIONS

In this paper, we introduced an innovative methodology for measuring latency in wireless-based communication networks, aiming to dissect the contributions of individual subsystem components, such as protocol stacks and WNICs. The proposed approach offers flexibility by enabling the placement of software probes at various points within the protocol stack, potentially allowing for the measurement of the latency introduced by multiple subsystems and components of the communication system.

To address subsystems that only permit a black-box measurement approach, like the WNIC, capture devices have been used that are capable of estimating latencies introduced by such systems. This approach strikes a balance between measurement accuracy, setup complexity, and cost.

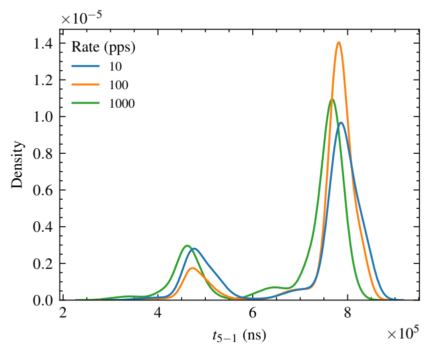
A critical aspect of the proposed methodology is the precise time synchronization the devices involved in the measurements need. This issue has been addressed by exploiting Time-Sensitive Networking, which is able to ensure that synchronization errors remain below tens of nanoseconds even in the worst-case scenarios.

A real-world use case has been addressed to investigate the proposed methodology. The results have shown that the approach offers a comprehensive understanding of latency dynamics within communication networks, offering insights into performance bottlenecks and facilitating targeted optimization efforts.

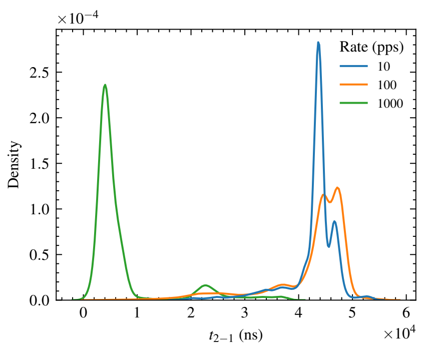
It is worth noting that, despite the proposed methodology having been tested with only two nodes in an isolated network, it can be easily extended to more complex scenarios. This includes networks with several nodes, multi-hop networks, whether public or private, and multi-domain networks (both wired and wireless). Accurate synchronization can still be achieved even in the presence of interfering traffic by utilizing additional features provided by TSN. Indeed, time-critical PTP traffic can be prioritized by reserving portions of bandwidth (IEEE 802.1Qav) or by scheduling and shaping traffic based on their priority class (IEEE 802.1Qbv).

Notably, in such complex scenarios, as a further issue, evaluating uncertainties becomes crucial, especially in multi-domain and multi-hop networks, as each domain and network segment may contribute individually to the overall uncertainty. Moreover, it is necessary to assess the potential limitations of the capture device, such as understanding the maximum amount of traffic it can actually receive and timestamp without introducing significant errors.

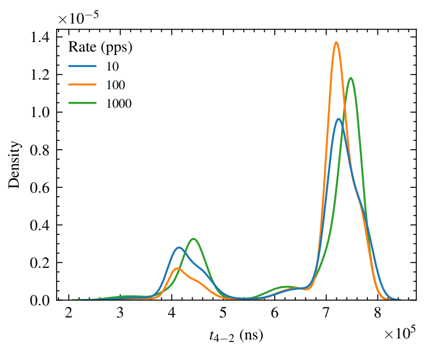
Moving forward, future research efforts will focus on characterizing the uncertainties inherent in the proposed measurement setup. Building upon this foundation, we aim to refine the measurement technique further and conduct additional tests, possibly



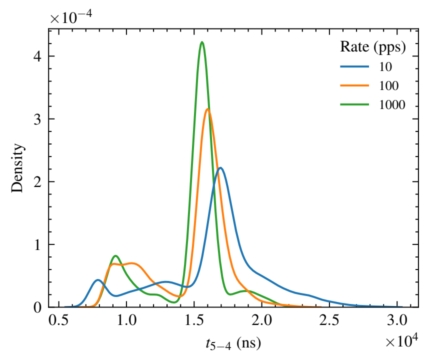
(a) EPDF of the end-to-end latency with unicast UDP packet stream.



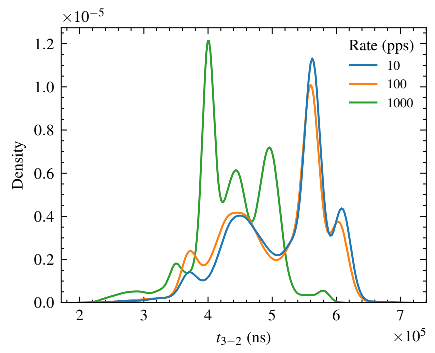
(b) EPDF of the transmission stack execution time with unicast UDP packet stream.



(c) EPDF of the WNIC latency in tx and rx and the transmission medium delay with unicast UDP packet stream.



(d) EPDF of the reception stack execution time with unicast UDP packet stream.



(e) EPDF of the WNIC latency during the transmission with unicast UDP packet stream.

Figure 8. EPDFs of different latencies under a unicast UDP packet stream.

increasing the number of probes involved in the measurements to enhance accuracy.

REFERENCES

[1] A. Amodei, D. Capriglione, M. Cheminod, P. Ferrari, G. Miele, A. Morato, F. Tramarin, S. Vitturi, E. Sisinni, C. Zunino, Time Sensitive Networking for Future Enabling Technologies: Overview and Measurement Issues for Metrological Characterization, 2024 IEEE International Symposium on Measurements & Networking (M&N), IEEE, Rome, Italy, Jul. 2024, pp. 1–6.
DOI: 10.1109/MN60932.2024.10615861

[2] A. Morato, E. Ferrari, S. Vitturi, F. Tramarin, A TSN-based Technique for Latency Measurement in Real-Time Wireless Communication networks, 2024 IEEE International Symposium on Measurements & Networking (M&N), IEEE, Rome, Italy, Jul. 2024, pp. 1–6.
DOI: 10.1109/MN60932.2024.10615590

[3] K. Nikhileswar, K. Prabhu, D. Cavalcanti, A. Regev, Time-Sensitive Networking Over 5G for Industrial Control Systems, 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, Stuttgart, Germany, Sep. 2022, pp. 1–8.
DOI: 10.1109/ETFA52439.2022.9921680

[4] A. Noguera Cundar, R. Fotouhi, Z. Ochitwa, H. Obaid, Quantifying the Effects of Network Latency for a Teleoperated Robot, Sensors (Basel, Switzerland), vol. 23, 2023, no. 20.
DOI: 10.3390/s23208438

[5] A. Amodei, D. Capriglione, G. Cerro, L. Ferrigno, G. Miele, G. Tomasso, A Measurement Approach for Inline Intrusion Detection of Heartbleed-Like Attacks in IoT Frameworks, IEEE Transactions on Instrumentation and Measurement, vol. 72, 2023, pp. 1–10.
DOI: 10.1109/TIM.2023.3282662

[6] L. Angrisani, A. Botta, G. Miele, A. Pescape, M. Vadursi, Experiment-Driven Modeling of Open-Source Internet Traffic Generators, IEEE Transactions on Instrumentation and Measurement, vol. 63, Nov. 2014, no. 11, pp. 2529–2538.
DOI: 10.1109/TIM.2014.2348633

[7] D. Fadhil, R. Oliveira, A Novel Packet End-to-End Delay Estimation Method for Heterogeneous Networks, IEEE Access, vol. 10, 2022, pp. 71 387–71 397.
DOI: 10.1109/ACCESS.2022.3188116

[8] N. Hariri, B. Hariri, S. Shirmohammadi, A distributed measurement scheme for internet latency estimation, IEEE Transactions on Instrumentation and Measurement, vol. 60, 2011, no. 5, pp. 1594–1603.
DOI: 10.1109/TIM.2010.2092871

[9] T. Fedullo, F. Tramarin, S. Vitturi, The impact of rate adaptation algorithms on wi-fi-based factory automation systems, Sensors, vol. 20, 2020, no. 18.
DOI: 10.3390/s20185195

[10] P. Ferrari, A. Flammini, S. Rinaldi, A. Bondavalli, F. Brancati, Experimental Characterization of Uncertainty Sources in a Software-Only Synchronization System, IEEE Transactions on Instrumentation and Measurement, vol. 61, May 2012, no. 5, pp. 1512–1521.
DOI: 10.1109/TIM.2011.2180974

[11] S. Sudhakaran, C. Hall, D. Cavalcanti, A. Morato, C. Zunino, F. Tramarin, Measurement method for end-to-end Time synchronization of wired and wireless TSN, 2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, Kuala Lumpur, Malaysia, May 2023, pp. 1–6.
DOI: 10.1109/I2MTC53148.2023.10176093

[12] M. Bosk, F. Rezabek, K. Holzinger, A. G. Marino, A. A. Kane, F. Fons, J. Ott, G. Carle, Methodology and Infrastructure for TSN-Based Reproducible Network Experiments, IEEE Access,

vol. 10, 2022, pp. 109 203–109 239.
DOI: 10.1109/ACCESS.2022.3211969

[13] T. Fedullo, A. Morato, F. Tramarin, L. Rovati, S. Vitturi, A Comprehensive Review on Time Sensitive Networks with a Special Focus on Its Applicability to Industrial Smart and Distributed Measurement Systems, Sensors, vol. 22, Feb. 2022, no. 4, p. 1638.
DOI: 10.3390/s22041638

[14] D. Cavalcanti, C. Cordeiro, M. Smith, A. Regev, WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11, IEEE Communications Standards Magazine, vol. 6, Dec. 2022, no. 4, pp. 22–29.
DOI: 10.1109/MCOMSTD.0002.2200039

[15] A. Al Sadi, D. Berardi, F. Callegati, A. Melis, M. Prandini, P4DM: Measure the link delay with P4, Sensors, vol. 22, 2022, no. 12.
DOI: 10.3390/s22124411

[16] X. Zhang, G. Min, Q. Fan, H. Yin, D. Oliver Wu, Z. Ma, A lightweight statistical latency measurement platform at scale, IEEE Transactions on Industrial Informatics, vol. 18, 2022, no. 2, pp. 1186–1196.
DOI: 10.1109/TII.2021.3098796

[17] A. Morato, C. Zunino, M. Cheminod, S. Vitturi, F. Tramarin, A TSN-based Technique for Real-Time Latency Evaluation in Communication Networks, 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, Glasgow, United Kingdom, May 2024, pp. 1–6.
DOI: 10.1109/I2MTC60896.2024.10560981

[18] L. Angrisani, D. Capriglione, L. Ferrigno, G. Miele, Measurement of the IP Packet Delay Variation for a reliable estimation of the mean opinion score in VoIP services, 2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings, May 2016, pp. 1–6.
DOI: 10.1109/I2MTC.2016.7520492

[19] K. Watabe, K. Nakagawa, Packet delay estimation that transcends a fundamental accuracy bound due to bias in active measurements, IEICE Transactions on Communications, vol. E100B, 2017, no. 8, pp. 1377–1387.
DOI: 10.1587/transcom.2016EBP3364

[20] N. Zhu, J. He, Y. Zhou, W. Wang, On the accuracy of packet delay estimation in distributed service networks, Journal of Network and Systems Management, vol. 21, 2013, no. 4, pp. 623–649.
DOI: 10.1007/s10922-013-9266-4

[21] M. Shahzad, A. Liu, Accurate and efficient per-flow latency measurement without probing and time stamping, IEEE/ACM Transactions on Networking, vol. 24, 2016, no. 6, pp. 3477–3492.
DOI: 10.1109/TNET.2016.2533544

[22] A. Carta, N. Locci, C. Muscas, F. Pinna, S. Sulis, GPS and IEEE 1588 synchronization for the measurement of synchrophasors in electric power systems, Computer Standards & Interfaces, vol. 33, Feb. 2011, no. 2, pp. 176–181.
DOI: 10.1016/j.csi.2010.06.009

[23] P. Pääkkönen, J. Prokkola, A. Lattunen, Instrumentation-based tool for latency measurements, Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering, ACM, Karlsruhe Germany, Sep. 2011, pp. 403–412.
DOI: 10.1145/1958746.1958802

[24] O. N. Pardo-Zamora, R. D. J. Romero-Troncoso, J. R. Millan-Almaraz, D. Morinigo-Sotelo, R. A. Osornio-Rios, Methodology for Power Quality Measurement Synchronization Based on GPS Pulse-Per-Second Algorithm, IEEE Transactions on Instrumentation and Measurement, vol. 70, 2021, pp. 1–9.
DOI: 10.1109/TIM.2020.3036691

[25] S. Caban, A. Disslbacher-Fink, J. A. Garcia-Naya, M. Rupp, Synchronization of wireless radio testbed measurements, 2011 IEEE International Instrumentation and Measurement Technology Conference, IEEE, Hangzhou, China, May 2011, pp. 1–4.
DOI: 10.1109/IMTC.2011.5944089

[26] P. A. Crossley, H. Guo, Z. Ma, Time synchronization for transmission substations using gps and ieee 1588, CSEE Journal of Power and Energy Systems, vol. 2, 2016, no. 3, pp. 91–99.
DOI: 10.17775/CSEEJPES.2016.00040

[27] L. Arceo-Miquel, Y. S. Shmaliy, O. Ibarra-Manzano, Optimal Synchronization of Local Clocks by GPS 1PPS Signals Using Predictive FIR Filters, IEEE Transactions on Instrumentation and Measurement, vol. 58, Jun. 2009, no. 6, pp. 1833–1840.
DOI: 10.1109/TIM.2009.2013654

[28] D. Pallier, V. Le Cam, S. Pillement, Energy-Efficient GPS Synchronization for Wireless Nodes, IEEE Sensors Journal, vol. 21, Feb. 2021, no. 4, pp. 5221–5229.
DOI: 10.1109/JSEN.2020.3031350

[29] Timestamping — The Linux Kernel documentation. Online. [Accessed: 2025-05-15].
https://docs.kernel.org/networking/timestamping.html

[30] R. Love, Linux System Programming, 2nd ed., O'Reilly, Sebastopol (Calif.), 2013, ISBN: 978-1-4493-3953-1.

[31] D. Macii, D. Fontanelli, D. Petri, A master-slave synchronization model for enhanced servo clock design, 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, IEEE, Brescia, Italy, Oct. 2009, pp. 1–6.
DOI: 10.1109/ISPCS.2009.5340199

[32] P. Tosato, D. Macii, D. Fontanelli, D. Brunelli, D. Laverty, A Software-based Low-Jitter Servo Clock for Inexpensive Phasor Measurement Units, 2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Sep. 2018, pp. 1–6.
DOI: 10.1109/ISPCS.2018.8543082

[33] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), Jul. 2008, pp. 1–269.
DOI: 10.1109/IEEESTD.2008.4579760

[34] N. Kasdin, T. Walter, Discrete simulation of power law noise (for oscillator stability evaluation), Proceedings of the 1992 IEEE Frequency Control Symposium, May 1992, pp. 274–283.
DOI: 10.1109/FREQ.1992.270003

[35] R. L. Scheiterer, C. Na, D. Obradovic, Gü. Steindl, Synchronization Performance of the Precision Time Protocol in Industrial Automation Networks, IEEE Transactions on Instrumentation and Measurement, vol. 58, Jun. 2009, no. 6, pp. 1849–1857.
DOI: 10.1109/TIM.2009.2013655

[36] R. C. Sofia, P. Mendes, C. J. Bernardos, E. Schooler, Requirements for Reliable Wireless Industrial Services, Jul. 2024. Online. [Accessed: 2026-01-01].
https://datatracker.ietf.org/doc/draft-ietf-detnet-raw-industrial-req

[37] F. Tramarin, S. Vitturi, M. Luvisotto, A Dynamic Rate Selection Algorithm for IEEE 802.11 Industrial Wireless LAN, IEEE Transactions on Industrial Informatics, vol. 13, Apr. 2017, no. 2, pp. 846–855.
DOI: 10.1109/TII.2016.2616327

[38] S. Vitturi, L. Seno, F. Tramarin, M. Bertocco, On the Rate Adaptation Techniques of IEEE 802.11 Networks for Industrial Applications, IEEE Transactions on Industrial Informatics, vol. 9, Feb. 2013, no. 1, pp. 198–208.
DOI: 10.1109/TII.2012.2189223

[39] G. Naik, D. Ogbe, J.-M. J. Park, Can Wi-Fi 7 Support Real-Time Applications? On the Impact of Multi Link Aggregation on Latency, ICC 2021 - IEEE International Conference on Communications, Jun. 2021, pp. 1–6.
DOI: 10.1109/ICC42927.2021.9500256

[40] M.-T. Suer, C. Thein, H. Tchouankem, L. Wolf, Comparison of Multi-Connectivity Schemes on Different Layers for Reliable Low Latency Communication, 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Sep. 2021, pp. 1357–1362.
DOI: 10.1109/PIMRC50174.2021.9569560