IEEE Open Journal of the
Industrial Electronics Society

Received 4 July 2024; accepted 27 July 2024. Date of publication 2 August 2024;
date of current version 14 August 2024. The review of this article was arranged by Associate Editor Luis Gomes.

Digital Object Identifier 10.1109/OJIES.2024.3436590

TALESS: TSN With Legacy End-Stations Synchronization

DANIEL BUJOSA MATEU [1] (Student Member, IEEE), JULIAN PROENZA [2] (Senior Member, IEEE), ALESSANDRO V. PAPADOPOULOS [1] (Senior Member, IEEE), THOMAS NOLTE [1] (Senior Member, IEEE), AND MOHAMMAD ASHJAEI [1] (Senior Member, IEEE)

[1] Mälardalen University, 72220 Västerås, Sweden
[2] University of the Balearic Islands, 07122 Palma, Spain

CORRESPONDING AUTHOR: DANIEL BUJOSA MATEU (e-mail: daniel.bujosa.mateu@mdu.se).

This work was supported in part by the Swedish Knowledge Foundation (KKS) through the FIESTA and SEINE projects, in part by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the DESTINE and PROVIDENT projects, and in part by XPRES project. The work of Julian Proenza was supported by MCIN/AEI/10.13039/501100011033/ERDF, EU, under Grant pid2021-124348ob-i00.

ABSTRACT In order to facilitate the adoption of Time Sensitive Networking (TSN) by the industry, it is necessary to develop tools to integrate legacy systems with TSN. In this article, we propose a solution for the coexistence of different time domains from different legacy systems, each with its corresponding synchronization protocol, in a single TSN network. To this end, we experimentally identified the effects of replacing the communications subsystem of a legacy Ethernet-based network with TSN in terms of synchronization. Based on the results, we propose a solution called TALESS (TSN with Legacy End-Stations Synchronization). TALESS can identify the drift between the TSN communications subsystem and the integrated legacy devices (end-stations) and then modify the TSN schedule to adapt to the different time domains to avoid the effects of the lack of synchronization between them. We validate TALESS through both simulations and experiments on a prototype. We demonstrate that thanks to TALESS, legacy systems can synchronize through TSN and even improve features such as their reception jitter or their integrability with other legacy systems.

INDEX TERMS Legacy support, synchronization, time-sensitive networking(TSN).

I. INTRODUCTION

Since the creation of the IEEE Time Sensitive Networking (TSN) Task Group (TG) in 2012, industry interest in TSN has continued growing. TSN seems to be essential for the incipient Industry 4.0 [1] as well as of interest in various areas such as automotive [2] and energy distribution [3]. The reason behind this growing interest is that TSN establishes a set of standards to provide deterministic zero-jitter and low-latency transmission, fault tolerance mechanisms, and advanced network management, allowing dynamic reconfiguration, precise clock synchronization, and flexibility in traffic transmission. The latter property is particularly relevant to the industry's adoption of TSN. This is so because the flexibility in the traffic transmission allows the transmission of different types of traffic over the same physical links, which enables the migration of all kinds of legacy traffic to TSN. Thereby, most legacy devices and implemented solutions could be kept, reducing adoption time and costs.

Furthermore, most current networks are composed of different subnetworks with different communication protocols to meet their specific requirements. This hinders communication between subnetworks and, therefore, their integrability. Moreover, this increases the complexity of the overall network due to the use of different technologies, cabling redundancy, etc. Thanks to the TSN's flexibility in traffic, it is possible to combine different types of traffic in the same network, facilitating the communication and integration of the subnetworks. This integration can be done in different ways, such as through the use of gateways. However, this would not allow subnetworks to take advantage of other TSN features, such as higher bandwidth or low jitter. Therefore, we propose to replace the communications subsystem of the legacy network

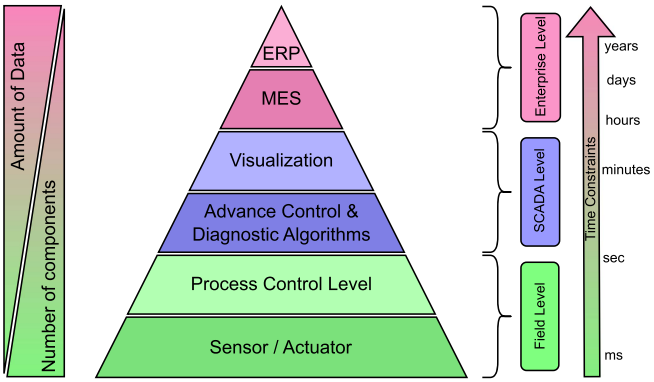

FIGURE 1. **Automation pyramid.**

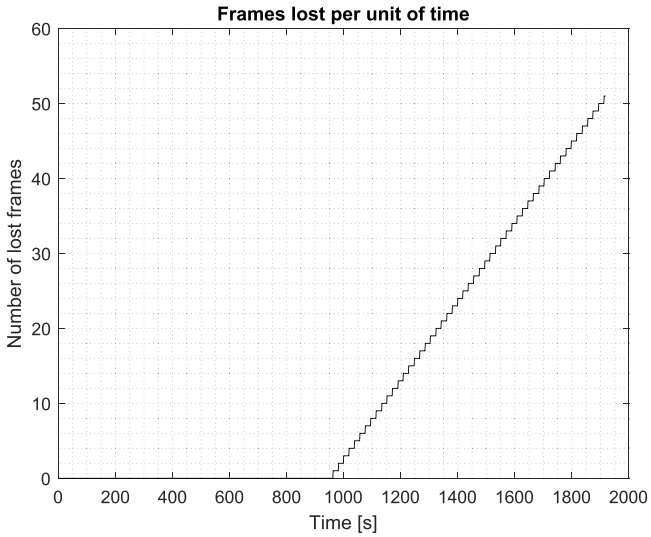

FIGURE 2. **Loss of frames per unit of time due to positive clock drift.**

directly, i.e., the set of devices exclusively responsible for communication, excluding the end-stations, with TSN but in such a specific way that the legacy end-stations can maintain their behavior and communication protocols (including their legacy synchronization protocol) agnostic to the change.

This approach improves the integrability of the different legacy systems and allows them to benefit from the advantages of TSN. For example, in recent years, the automotive industry has witnessed a substantial surge in complexity, driven by growing interest in vehicle automation. This complexity extends to the embedded networks within vehicles, as an increasing number of devices require increasing exchange of information with diverse and demanding timing requirements. Modern vehicles can now integrate hundreds of devices across various communication networks, with certain devices featuring multiple output ports for coordinating actions across different network layers like multimedia, data transmission, and comfort control. Integrating these networks through TSN offers a solution to reduce system complexity significantly. This integration streamlines manufacturing and maintenance processes and facilitates the addition of new devices to enhance functionality. Other benefits could be weight reduction and access to TSN advantages for both new and legacy devices, including fault-tolerance mechanisms and zero-jitter reception. This transformation paves the way for achieving unprecedented levels of automation and meeting previously unattainable requirements for legacy systems. However, certain types of TSN traffic, such as Time-Triggered (TT) traffic, require the path from the source to the destination, including all switches in the network, to be synchronized to have a common time view. This is because TT traffic is transmitted according to a fixed time schedule that needs to be known and respected by all network components. This requirement is not fulfilled in many industrial networks where non-TSN nodes (*end-stations* in TSN terminology) do not feature TSN synchronization mechanisms and may be unable to support them due to their hardware or software limitations.

For instance, in the automation pyramid depicted in Fig. 1 [4], it can be seen how it is composed of the field, the supervisory control and data acquisition, and the enterprise levels. Each of these levels entails distinct temporal constraints, data transmission volumes, and varying numbers of components, among other factors. Traditionally, this has been addressed using several networks with different characteristics connected through gateways with limited internetwork connectivity like PROFINET [5], EtherCAT [6], or Sercos [7] (Industrial Ethernet) for real-time traffic at the lower levels of the pyramid and Ethernet for higher levels of the pyramid. However, thanks to TSN, this separation is no longer necessary, as a single network can handle all types of traffic. Nonetheless, in an established factory, the cost-effectiveness of replacing all end-stations with TSN end-stations may not be cost-effective. In addition, each subnetwork mentioned above employs a distinct synchronization protocol, none of which may be the Generalized Precision Time Protocol (gPTP) [8] used in TSN. Moreover, a recent study has quantified, in terms of loss of transmitted frames, the impact caused by mixing in the same network components that use different synchronization protocols [9]. Fig. 2 depicts the number of frames lost over time in a heterogeneous network, wherein legacy end-stations synchronized through a legacy synchronization protocol other than gPTP communicate over a TSN network. These results highlight the need for a mechanism capable of harmonizing the different synchronization mechanisms that can coexist in a heterogeneous TSN network.

For the remainder of the article, we will use the term *legacy network* for each original network, i.e., before replacing its communications subsystem with TSN. On the other hand, the term *legacy end-stations* will be used in reference to any node of any legacy network that has been integrated by replacing its communications subsystem with TSN, while the term *legacy system* will refer to the set of end-stations that were initially part of the same legacy network and are synchronized through its own legacy synchronization protocol, thus sharing a common time view. Fig. 3 illustrates the terminology introduced

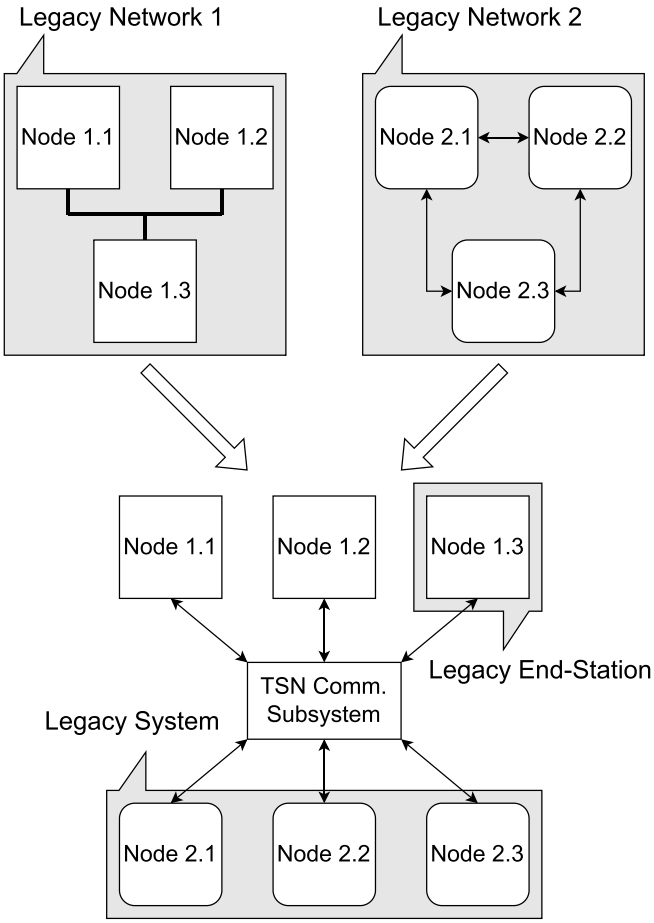

FIGURE 3. Terminology used in the integration of legacy networks in TSN. Integration of two legacy networks, legacy network 1 with a bus topology and legacy network 2 with a ring topology, into a single TSN heterogeneous network combining two legacy systems consisting of the legacy end-stations from the two legacy networks.

with a diagram. It shows how two *legacy networks*, one with bus topology (Legacy Network 1) and one with ring topology (Legacy Network 2), are integrated into a single TSN network. This network is formed by the TSN communications subsystem and by two *legacy systems*, which in turn are created by the set of *legacy end-stations* from the *legacy networks*.

The critical piece for achieving the above-indicated integration of the legacy end-stations with the new TSN communication subsystem is a novel mechanism we propose in this article. This mechanism is called TALESS (TSN with Legacy End-Stations Synchronization), and it is devised to prevent the negative effects resulting from the lack of synchronization between the TSN communications subsystem and the legacy systems integrated with it. TALESS transparently improves network performance without requiring modifications to legacy systems. To achieve this, rather than synchronizing the end-stations with TSN, which would necessitate modifications to the software or hardware of these devices, we tailor the TSN schedule to accommodate the unique time domains of each legacy system integrated into the TSN network. As we

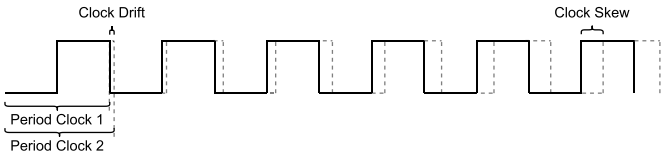

FIGURE 4. Clock drift and clock skew.

will see in Section IV, the main consequence of the lack of synchronization is clock drift, which is the root cause of the adverse effects. Clock drift causes two clocks to progress at different rates, leading to a clock skew, i.e., an accumulated discrepancy over time [10]. Fig. 4 visually illustrates this behavior. Regarding frame transmission and reception, traffic drift refers to the variance between a frame's real transmission or reception period and its scheduled one, whereas traffic skew denotes the temporal disparity between the expected or scheduled reception time of a frame and its actual reception time. In this context, TALESS does not aim to remove the drift between the TSN clock and the legacy system clocks, but rather to eliminate the effects of this drift by adapting the TSN schedule.

This approach allows legacy systems to maintain their distinct time domains while benefiting from the enhancements offered by TSN, ensuring seamless operation without compromising its previous functionality and potentially improving it. Preventing any modifications in the legacy end-stations makes TALESS a general solution that allows applying the proposed integration approach on any TSN network where several Ethernet-based legacy systems communicate with different communication protocols. On the one hand, we validate it using a model that simulates long executions (1 year) of a communication network. This model simulates the behavior of the TSN network with and without TALESS for different types of legacy end-station transmissions. On the other hand, we implement TALESS in a network prototype by which we experimentally validate the solution and verify both the implementation and its simulation model. However, in our experiments, we amplified certain clock parameters of the legacy system to magnify the effects of the solution and thereby be able to demonstrate its behavior in a reasonable run-time.

Contributions. As indicated, among the different requirements for integrating legacy systems into a TSN network, an essential aspect is clock synchronization. Maintaining proper communication behavior among devices is needed, especially if these devices require TT traffic transmissions. Thus, the main target of this article is to develop a mechanism to avoid the adverse effects of carelessly putting together legacy systems with TSN in terms of clock synchronization. The main contributions of this article are as follows.

1) We identify problems caused by the lack of synchronization through experiments on a network prototype.
2) We propose a mechanism named TALESS to remove the effects of lack of synchronization when including legacy systems into a TSN network.

3) We model TALESS to validate the effectiveness of the proposed solution in a simulation environment with realistic network values.
4) Finally, we implement TALESS in a network prototype to experimentally showcase its impact on utilizing legacy systems in a TSN network. We also compare the experiment results with the simulation model to verify the implementation and the simulation.

Outline. The rest of this article is organized as follows. Section II presents the related work. Section III provides the necessary background to understand this article better. Section IV presents the effects of carelessly including legacy systems into a TSN network in terms of clock synchronization. Section V proposes TALESS. Section VI presents the simulation model and experimental setup used to validate TALESS, while Section VII presents the results obtained from both the simulations of the model and the experiments on the prototype. Finally, Section VIII concludes this article.

II. RELATED WORK

One of the most crucial aspects of TSN technology is clock synchronization. However, to our knowledge, no work has provided a solution to the adverse effects caused by the lack of synchronization in heterogeneous TSN networks that combine one or more Ethernet-based legacy systems through a TSN communications subsystem. On the contrary, most studies aim to integrate TSN with wireless and 5 G networks. For example, a low-overhead beacon-based time synchronization method was implemented to provide precise synchronization in wireless networks in highly deterministic TSN networks, as outlined in [11]. Other research has focused on extending IEEE 802.1AS and IEEE 802.11 to enable TSN integration with wireless networks, as described in [12] and [13]. The challenges of integrating Wired TSN and WLAN technologies and a possible solution in a hybrid TSN device architecture were discussed in [14]. Moreover, the study in [15] presented TSN clock synchronization that aligns with 5 G specifications. To solve cross-domain clock synchronization issues in 5G-TSN networks, a method based on data packet relay was proposed in [16]. Finally, the performance of 5G-TSN networks was also evaluated in terms of clock synchronization in several works such as in [17], [18], and [19].

On the other hand, limited research explores synchronization in heterogeneous TSN networks, i.e., networks that incorporate TSN and non-TSN devices. For instance, Xue et al.[20] presented a method for preserving synchronization across TSN subnetworks connected through non-TSN switches. Their approach involves estimating the delays experienced by the synchronization messages passing through these devices and configuring TSN networks to minimize these delays. In contrast, our work focuses on integrating legacy systems into a single TSN network. Notably, there are even fewer studies addressing synchronization between legacy end-stations and TSN. Bujosa et al. [21] presented one methodology for integrating EtherCAT and TSN in terms of clock synchronization. However, this type of integration requires customized solutions for each integrated protocol, which can pose a challenge to the broader adoption of TSN by the industry. This is because designing and implementing these solutions take significant time and resources, and compatibility between solutions can also be demanding. Note that our approach differs from these solutions. In our proposed TSN heterogeneous networks, legacy systems are not synchronized with TSN, as they operate on distinct synchronization protocols. Instead, TSN adjusts to the clock timing of legacy systems to mitigate the negative impacts of the lack of synchronization.

Regarding the integration of legacy systems, several papers have proposed solutions for integrating TSN with different proprietary field buses. For example, Szancer et al. [22] proposed a migration method for SERCOS III into TSN. However, as the synchronization mechanisms of both protocols are incompatible, the authors opted to adopt TSN's gPTP on SERCOS III devices. Furthermore, Nsaibi et al. [23] also on integrating SERCOS III over TSN, limited the synchronization and integration to be only between the master and the TSN network, leaving the slaves disconnected to TSN. Regarding the integration of TSN with PROFINET, in Schriegel and Jasperneite [24], the authors proposed a new type of switch that allows the mapping of PROFINET traffic on TSN. However, it does not prevent clock drift between TSN and PROFINET end-stations with the possible adverse effects that this would entail. Finally, Barzegaran et al.[25] and Zhao et al. [26] introduced TSN schedulers designed for unscheduled and unsynchronized legacy traffic exhibiting high jitter. However, they are unable to guarantee zero-jitter reception and do not consider clock drift. Both studies operate under the assumption of a constant period for legacy frames. Nevertheless, due to drift, this assumption may not hold true from the perspective of TSN. All these TSN integration works with different legacy systems could benefit from the solution proposed in this work since synchronization limitations would be avoided.

In a work presented by Bujosa et al.[9] implemented a non-TSN network with its own synchronization protocols and replaced its communications subsystem with TSN. The work preliminary identified the effects of the lack of synchronization between the legacy system and the TSN network due to the lack of integration between the synchronization protocols used by the legacy system and the TSN's gPTP. Through several experiments, authors detected the causes and consequences of the network's lack of synchronization in the short and long term. However, the work was a short paper that merely suggested uncertain and indeterminate solutions that lacked implementation and proper validation.

III. BACKGROUND

In TSN networks, communication between end-stations is achieved by transmitting Ethernet frames along Ethernet links and TSN switches. In TSN switches and end-stations, each output port has up to 8 FIFO queues, each corresponding to one specific priority level. TSN frames are assigned to one of

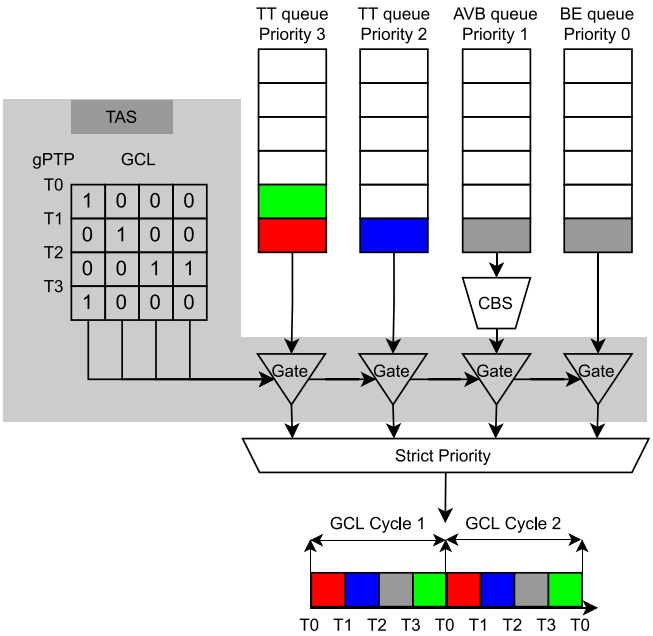

FIGURE 5. A TSN egress port with four FIFO queues: two TT queues, one AVB queue, and one BE queue.

the 8 priorities, or queues, configured as one of the three types of TSN traffic, including TT, Audio Video Bridging (AVB), and Best-Effort (BE) traffic. TT traffic is commonly given the highest priority, while BE traffic has the lowest priority. Several queues can be configured as the same type of traffic, thus giving different classes, for example, AVB class A, B, and C. An illustration of these concepts can be seen in Fig. 5, which shows a TSN device (either an end-station or switch) output port with four queues configured to convey two TT traffic classes with the highest priority, one AVB traffic class with medium priority, and BE traffic with the lowest priority. As we will discuss later, TT traffic relies on the Time Aware Shaper (TAS) for transmission isolation, ensuring zero blocking and interference, resulting in the transmission according to the schedule with zero jitter. In contrast, AVB traffic utilizes both TAS to avoid blocking TT traffic and CBS to restrict the maximum bandwidth for each AVB queue, improving lower priority queues' quality of service. Last, BE traffic also utilizes TAS, hence it can transmit only when TT traffic is not transmitting and after AVB traffic has utilized its allocated bandwidth, since it has the lowest priority.

Next, we explain three critical aspects of the background for this work. First, we will introduce the TAS and the gPTP since they are the main mechanisms responsible for TT transmission, which is the type of traffic most affected by the lack of synchronization. On the other hand, we will explain the centralized network configuration (CNC) element, a key component for TALESS implementation. We will not delve deeper into CBS and the other traffic classes (AVB and BE) as they are not relevant to this study, given their synchronization-independent operation.

A. TIME AWARE SHAPER

To provide the determinism required by TT traffic and, therefore, to know exactly when each TT frame is transmitted, TSN must be able to prevent inter-frame interference. To do this, TSN uses the TAS mechanism shown in Fig. 5. This mechanism assigns a gate to each queue that can be open or close. The state of the gate is determined by the gate control list (GCL), which specifies at the nanosecond level how long a gate should be open or closed in a cyclically repeating list. If the gate of a queue is open, it can transmit the traffic in the queue. Otherwise, the frames in that queue are blocked from transmission. The opening period of a gate is called a *transmission window* or simply a *window*.

The operation of TAS for two TT queues is also depicted in Fig. 5. In this example, three TT frames with a period of 4 time units and transmission time of 1 time unit are transmitted through a TSN switch port, where two of the frames are assigned the highest priority 3 (green and red), and one frame (blue) is assigned priority 2. The hyper-period, the least common multiple of the frames' periods, is calculated to schedule the transmissions. This value is used to define the GCL cycle, which controls the transmission of the frames by specifying the open or closed state of the gates associated with each priority queue. Thus, the GCL cycle in this example is set to 4 time units; hence, the list will be repeated every 4 time units. From time T0 to T1, the gate for priority 3 queue is open, allowing the transmission of the red frame, while the gate for the other queues remains closed. From T1 to T2, the blue frame, which has priority 2, can be transmitted as its gate is open. Both gates are closed between T2 and T3, resulting in no TT transmission but allowing lower priority queues to transmit even if higher priority frames are waiting for transmission. Finally, the gate for the priority 3 queue is open in the last transmission window, allowing the transmission of the green frame. The bottom of Fig. 5 displays two cycles of frame transmissions, which shows the repetition of the GCL list.

B. GENERALIZED PRECISION TIME PROTOCOL

The mechanism providing the TSN clock synchronization (gPTP) is described in the IEEE 802.1AS standard. It consists of three main parts, including the best master clock algorithm (BMCA), the propagation delay measurement (PDM) mechanism, and the transport of time-synchronization information (TTI). BMCA determines the grandmaster clock, which is the reference clock in the TSN network, and the hierarchy between the different TSN devices. The PDM mechanism is used once the hierarchy is established to measure the propagation delay between systems. Finally, the TTI mechanism is used to forward the grandmaster time, which, together with the measured propagation delay, synchronizes the other TSN devices updating their internal clocks.

This synchronization protocol can achieve a clock accuracy of tens of nanoseconds. However, it has stringent software and especially hardware requirements that, in most cases, legacy

devices from Ethernet-based networks cannot support. First, the absence of gPTP implementation in legacy devices poses a challenge, as modifying these devices implies high costs. In addition, TSN requires network interfaces with hardware clocks capable of timestamping transmission and reception times, a feature lacking in most legacy devices. Even if legacy device software were modified to integrate gPTP, space constraints and hardware limitations would pose significant obstacles, potentially requiring the replacement of network interfaces and additional resources.

Furthermore, legacy systems employ diverse synchronization protocols. While the network time protocol (NTP) [27], as a precursor of gPTP, shares similarities with it in functionality, other protocols like EtherCAT [6] or flexible time-triggered (FTT) [28] utilize unique mechanisms unrelated to TSN. Modifying synchronization mechanisms in such cases would not only affect synchronization, but also demand system-wide overhauls, potentially impacting application-level implementations. Hence, an independent synchronization mechanism for legacy end-stations is necessary for TSN adoption in legacy networks.

C. CNC ELEMENT

The CNC is a virtual component that can be placed in a designated node, an end-station, or a switch. Regardless of its placement, it can exchange information with network devices via NETCONF [29], [30]. This bidirectional communication allows end-stations to send user or network configuration requests to the CNC while switches can communicate their specifications. Finally, the CNC can distribute new configurations to the entire network.

NETCONF utilizes a client–server approach for configuring the network, where the CNC acts as the client, responsible for collecting network information and initiating network device configurations. Note that all TSN network devices, e.g., TSN switches, must have a NETCONF server enabled to receive configurations from the CNC.

IV. PROBLEM STATEMENT

To observe the problems caused by the lack of synchronization between legacy systems and the TSN communication subsystems, we set up a small legacy network consisting of two single-board computers, i.e., Raspberry Pi (RPi) 3 Model B, running RPi Operating System (OS), connected point-to-point. Afterward, we add a Multiport TSN kit switch from System-on-Chip Engineering (SoC-e)[1] so that the RPIs behave as legacy end-stations in the new network, see Fig. 6. The Raspberry Pi boards are configured to synchronize their software clocks with each other via NTP. Note that any clock synchronization protocol other than gPTP could be used between the legacy end-stations since they reproduce scenarios, where the TSN switch cannot synchronize with the end-stations. We use NTP as a possible synchronization algorithm

FIGURE 6. Heterogeneous TSN network with legacy end-stations topology.

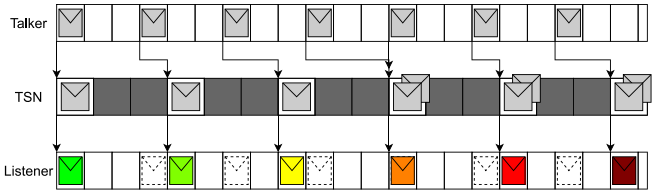

FIGURE 7. Positive legacy system clock drift behavior.

even if it is more common in industry the use of PTP, which typically provides a better synchronization accuracy.

In this experiment, we analyze the legacy network separately, i.e., without the TSN network, to see its baseline behavior. Then, a TSN network is added to the legacy system to analyze the effects of putting both together. These experiments show that the only traffic affected by the lack of synchronization is the scheduled traffic; therefore, it is the one we will focus on in this article. In this regard, thanks to the improved hardware and software capabilities of the TSN switches, the jitter of the legacy network TT traffic practically disappears. The reduced reception jitter would improve the system specifications and capabilities, enabling better service provision. Such enhancements would be challenging to achieve with the limitations of the communications subsystems previously used in the legacy network. However, due to the lack of synchronization, there is a drift between the clock time of TSN and the legacy system. This causes a deviation between the communication schedule of the legacy system and the TSN schedule that can be either positive or negative depending on which clock is faster or slower.

Regarding legacy synchronization, different protocols may require different configuration approaches. Traditional methods involve configuring synchronization traffic as AVB traffic to cap maximum latency or as TT traffic via the TAS for periodic configuration traffic. Alternatively, less conventional strategies like allocating a high-priority queue solely for synchronization traffic may be required. Nevertheless, these unconventional methods might compromise the maximum jitter experienced by TT traffic due to potential interference from synchronization traffic. Nonetheless, this jitter is expected to remain lower than that of the legacy network. However, these specific solutions fall beyond the scope of this article. Below, we explain the findings of the experiment in detail.

Fig. 7 shows the behavior of a heterogeneous TSN network in which the legacy system experiences a positive clock

[1]MTSN Kit: a Comprehensive Multiport TSN Setup. [Online]. Available: https://soc-e.com/mtsn-kit-acomprehensive-multiport-tsn-setup/

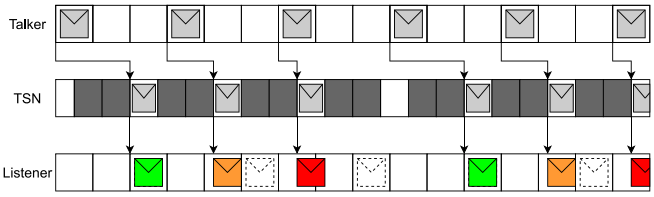

FIGURE 8. Negative legacy system clock drift behavior.

drift relative to the TSN communication subsystem. When the TSN clock is slower than the legacy system clock, the legacy system schedule exhibits a positive drift relative to the TSN schedule, causing frames to arrive at the receiver increasingly later than their legacy scheduled time. Moreover, since the transmission of frames by the TSN network to the legacy system receiver (*listener* in TSN terminology) is slower than the transmission by the legacy system transmitter (*talker* in TSN terminology) to the TSN network, the frames stack up in the buffers. However, the buffers are not infinite. Hence, frames that arrive once the buffer is full are discarded.

Fig. 2 shows the number of frames lost (y-axis) per time unit x-axis during the experiment in which the legacy system experiences a positive clock drift (D) relative to the TSN communication subsystem. This experiment demonstrates that after a period of frame accumulation in the output queue, the queue starts to lose one frame out of every 100/(D [%]) frames.

Fig. 8 shows the behavior of a heterogeneous TSN network in which the legacy system experiences a negative clock drift relative to the TSN communication subsystem. When the TSN clock is faster than the legacy system clock, the legacy system schedule exhibits a negative drift relative to the TSN schedule, causing frames to arrive at the receiver increasingly earlier than their legacy scheduled time. However, this effect cannot be infinitely extended over time since receiving a frame before it has been transmitted is impossible. When enough clock skew accumulates after a while, frames miss the transmission window in which they are scheduled, leaving a period with no frames being transmitted.

Fig. 9 shows the traffic skew observed during the experiment where the legacy system encounters a negative clock drift relative to the TSN communication subsystem. In this scenario, the loss of transmission windows becomes evident through the abrupt shifts in traffic skew observed in the graph. These findings reveal that frames undergo an entire period of clock drift before losing the transmission window and resetting the drift.

Through these experiments, which presented similar results to those in [9], we can observe that legacy systems can continue communicating through TSN and benefit from some of its features, such as improved reception jitter. However, due to the lack of synchronization, a clock drift appears, which not only causes a deviation in reception, but can lead to empty transmission windows or even loss of frames. Therefore, this work aims to develop a mechanism that eliminates the drift

FIGURE 9. Traffic skew due to negative clock drift.

between the TSN and legacy system schedules without requiring any modification in the legacy end-stations.

V. TALESS: TSN WITH LEGACY END-STATIONS SYNCHRONIZATION

As we have discussed, drift is the primary source of errors in the absence of synchronization. However, when it comes to developing a solution, we must consider two crucial factors. First, the lack of synchronization among various legacy systems, each operating in its unique time domain, can lead to different drifts with respect to the TSN network. Second, these drifts are not constant over time, as environmental factors like temperature can impact the clocks in the network differently. Therefore, the proposed solution should eliminate the clock drift effects of different legacy systems that change over time.

One way to avoid the negative consequences of the drift caused by the lack of synchronization consists in eliminating the drift between the TSN network schedule and the legacy system rather than among clocks. As discussed in the previous section, the clock drift between the legacy system and the TSN communication subsystem leads to a disparity between the rates of frame reception and forwarding in the switches. When forwarding lags behind reception, a buffer overflow may occur, while faster forwarding than reception results in the loss of transmission windows, causing delays of nearly two periods between consecutive frames. Ensuring that the frame forwarding rate matches the reception rate through proper scheduling would solve these issues.

To achieve this, we propose to modify the size of the TSN GCL transmission windows when there is drift. This way, we can modify the TSN's transmission pace to match the legacy system one. Fig. 10 shows an example of the operation of the proposed solution. This figure shows how, after detecting the drift, the TSN network changes the size of certain windows, specifically the lower priority BE queue, so that from that

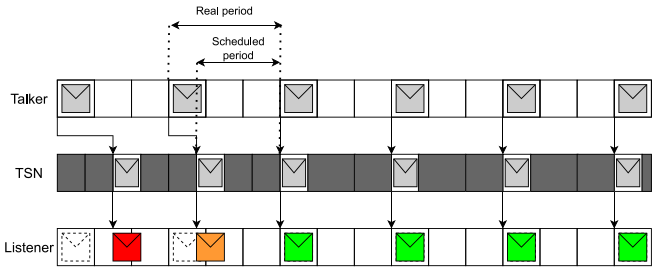

FIGURE 10. **TALESS operating diagram.**

point onward, the frames arrive at the receiver according to the legacy system schedule. However, the TT traffic transmission windows should not be modified since the size of these windows is determined by the size of the frame and the link speed, where both parameters are independent of the clock drift. In this regard, TALESS would have no negative effect on any critical traffic unless a network reaches 100% utilization and the legacy system's clock becomes faster with respect to the TSN network. In this case, reducing any transmission windows would cause adverse effects on the network since there would not be sufficient resources in the TSN network. However, configuring to 100% utilization on the network is impractical, and industrial use cases commonly avoid that. Therefore, in TALESS, non-TT windows (NTTW) should be modified by a ratio equal to the drift between the legacy system and TSN (D) plus the cumulative variation in TT windows (TTW). Therefore, the new size of each NTTW ($\text{NTTW}_i.\text{size}'$) can be computed as follows:

$$\text{NTTW}_i.\text{size}' = \text{NTTW}_i.\text{size} + D \times \text{NTTW}_i.\text{size} + D \times (\text{NTTW}_i.\text{start} - \text{NTTW}_{i-1}.\text{end}) \quad (1)$$

where to the previous NTTW size $\text{NTTW}_i.\text{size}$, we first add the variation of the window by multiplying the previous size $\text{NTTW}_i.\text{size}$ by the drift percentage D (either positive or negative) and second we add the cumulative variation of the TTW between the previous NTTW NTTW_{i-1} and the current one. This last increment is because TTW cannot be modified, and the increment of these windows accumulates until the next NTTW. Note that the GCL is a list of transmission windows that specifies the size of each window. The start of each window is determined by the size of the windows preceding it. Consequently, while TTWs maintain a fixed size, they can be shifted forward or backward based on adjustments to the NTTW according to (1). Moreover, by exclusively adjusting the size of the NTTW windows, inherent rescheduling issues such as overlapping can be avoided. Fig. 11 illustrates how modifying the NTTWs allows for the adjustment of the periods of the TTWs to align with the drift of legacy end-stations. The figure shows how TALESS adjusts the size of the NTTWs to align the periods of two TT frames initially set at 3 and 6 time units, respectively, to accommodate end-stations with approximately ±8 % drift. Revisiting Fig. 10, we can observe that the implementation of the (1) results in the expansion

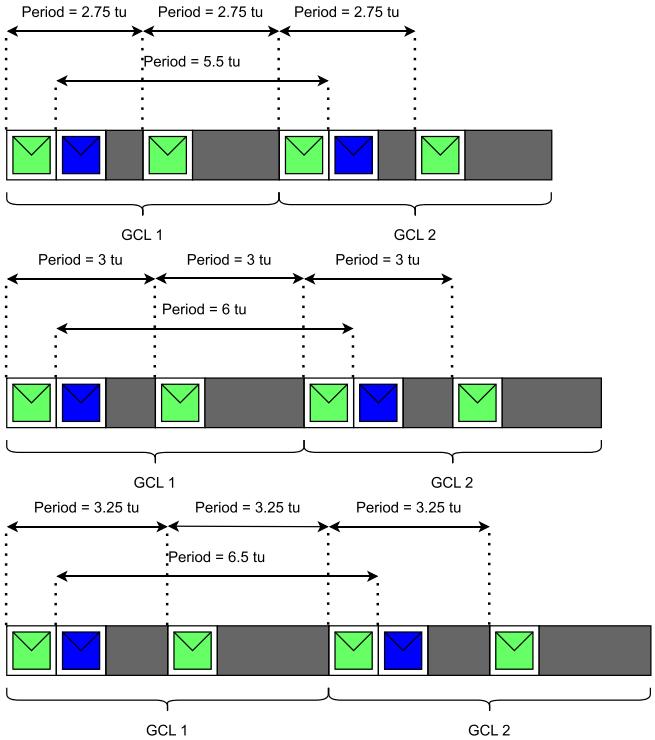

FIGURE 11. **Adjustment of the periods of 2 TTWs with 3 and 6 time units period (green frame and blue frame respectively) by modifying the size of NTTW (gray boxes) to accommodate end-stations' traffic with approximately ±8 % drift.**

of the gray transmission windows, which correspond to the NTTWs, allowing them to match the transmission pace of the legacy system. Moreover, the NTTWs located after a TTW exhibit a longer extension due to their assimilation of the expansion corresponding to the TTW.

Consider a single TT frame with a size of 1 time unit and a period of 4 time units, forming the GCL of a TSN switch with a TTW of 1 time unit and an NTTW of 3 time units. Assuming a -10% drift of the legacy system relative to the TSN clock (i.e., D = -0.1), each TSN time unit corresponds to 0.9 time units of the legacy system.

Over 10 cycles of the GCL, there would be 10 transmission windows, but the legacy transmitter would have sent $(10 \times 4)/(4 \times 0.9) = 11$ frames due to the time conversion, resulting in an accumulation of one frame. With 20 cycles, it would accumulate 2 frames, with 30 cycles, 3 frames, etc., leading to a buffer overflow. However, applying the proposed solution, the resulting GCL would have a TTW of 1 time unit and an adjusted NTTW of $3 - 0.1 \times 3 - 0.1 \times 1 = 2.6$ time units. Consequently, regardless of the number of GCL cycles n, the number of transmission windows and legacy talker transmissions would remain equal $n \times (1 + 2.6)/4 \times 0.9 = n$.

Equation (1) would be sufficient in a heterogeneous network where the drift between the legacy system and the TSN network is constant. In that case, it would be enough to calculate the drift and apply the formula to the TSN schedule only

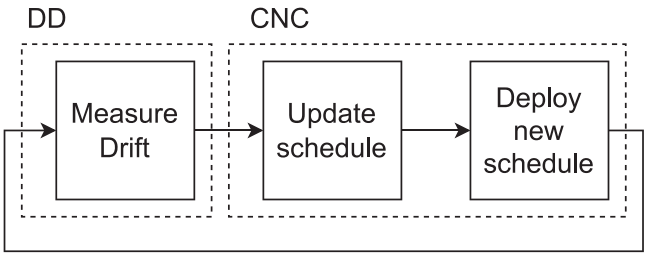

FIGURE 12. TALESS task flow.

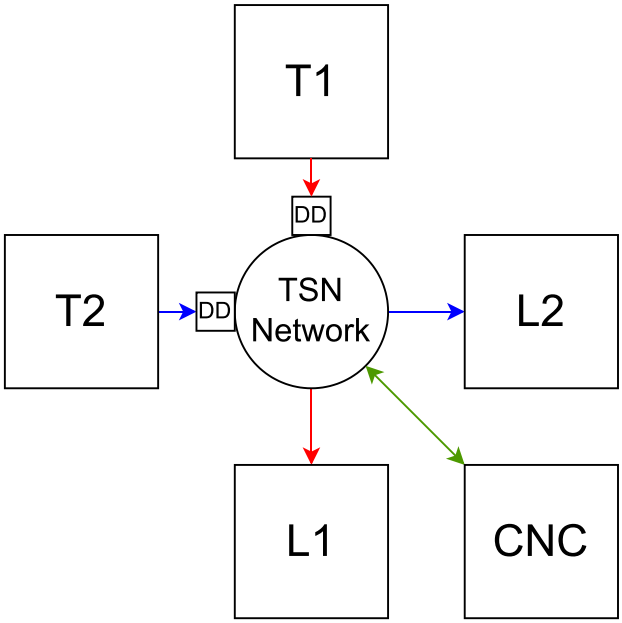

FIGURE 13. TALESS architecture.

once offline. Drift can be measured by sampling the network traffic and comparing the real periodicity with the scheduled one. However, the previous solution will not be sufficient if the drift is variable or if several legacy systems with different time domains coexist in the same TSN network. Regarding variable drifts, constant monitoring, and reconfiguration of the network is necessary. To do this, we propose a Drift Detector (DD) that continuously detects the drift between different clocks during run-time. Thus, we propose implementing a reconfiguration mechanism in the CNC, as shown in Fig. 12. The DD, located on at least one reception port of a switch connected to a legacy system talker, samples the reception, i.e., the legacy system talker transmission. A single transmitter suffices because, under the assumption that the entire legacy system is synchronized using a legacy synchronization protocol, the drift between all legacy end-stations and the TSN communication subsystem is the same. The diagram in Fig. 13 depicts a network that implements TALESS, in which end-stations T1 and T2, as well as L1 and L2, represent the talkers and listeners of legacy systems 1 and 2, respectively. The drift is determined based on the reception times of talker transmissions, which can be calculated using various methods. While this article does not aim to provide the optimal or most efficient method, some options are outlined as follows.

The simplest approach involves utilizing a periodic frame with easily identifiable characteristics. By sampling the reception times, it becomes possible to compute the time interval between consecutive receptions of this periodic frame. Statistical inference techniques, such as *T*-Student analysis, can then be applied to ascertain, with a user-defined confidence level, whether the sampled frame adheres to the intended periodicity established in the TAS schedule within the TSN switch. The user-defined confidence level sets the threshold for drift detection by the DD. Specifically, in a heterogeneous TSN network incorporating several legacy systems, one transmitter is selected from each legacy system, and a TT frame is chosen from each of the selected transmitters. These frames can be, for example, periodic transmissions from any type of sensor such as temperature, pressure, revolutions, etc., or a combination thereof. Subsequently, a DD responsible for sampling the selected TT frames is deployed on the reception port of each TSN switch connected to the selected transmitters. Each DD knows the period of the corresponding frame since the TSN switch schedules the TT frames and therefore knows their periodicity. In this example, we assume a scheduled period of 1 s. Once the network is operational, any drift between the legacy systems and TSN may commence due to the lack of synchronization. Consequently, the sampled frames may arrive with an average period different than the 1 s expected. For instance, one of the sampled frames may arrive with a 0.9 s period. By continuously comparing this average period (0.9 s) with the scheduled value (1 s), the DD can identify the presence of drift. Notably, this comparison entails statistical inference, owing to reception jitter. Therefore, establishing a threshold for statistical inference becomes essential, whether it be a 90%, 95%, or 99% confidence level. This confidence level, in conjunction with the network jitter and drift, dictates the maximum achievable traffic skew before drift detection. The determination of this maximum traffic skew is driven either by user specifications or network requirements. For example, reducing the statistical confidence level may become necessary in systems with stringent delay and jitter requirements. This adjustment could lead to more false positives in drift detection, prompting additional reconfigurations. However, it ensures that traffic skew remains within acceptable limits set by jitter and delay constraints. Section VII presents examples illustrating the maximum skew detected in the experiments conducted in this article and provides a detailed explanation of the calculation process.

Every time the drift is detected, the average period of the last "*n*" receptions of the sampled frame is calculated and divided by the expected frame period in the TSN switch to determine the drift percentage. This method was employed in the experiments discussed in Section VI.

Alternatively, other methods may involve calculating the reception rate per time unit. For instance, if the TSN switch expects to receive two TT frames from the talker, one with a

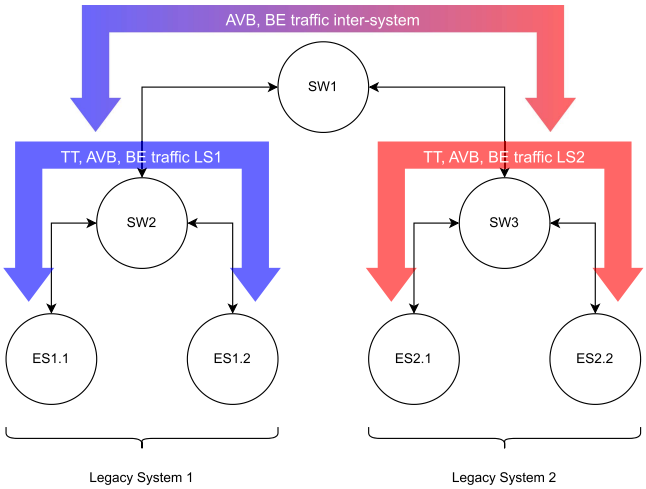

FIGURE 14. **Heterogeneous TSN network with tree topology using TALESS.**

period of 2 time units and the other with a period of 3 time units, it should ideally receive a total of 5 frames within every 6 time units interval. Through variations in the reception rate, it is possible to determine the drift. However, these methods are less accurate and require longer analysis periods to complete the determination.

Whenever a new drift is detected, a signal is sent to the CNC informing about the drift value. The CNC then updates the network configuration according to (1) and deploys it on the network to eliminate the drift between the legacy end-stations and the TSN schedule.

Finally, to allow the solution to work in networks combining different legacy systems, the only requirement is that TT traffic routes of different legacy systems cannot share output ports. This is because variations between the drifts of the legacy systems would invalidate TSN scheduling since the different drifts could cause some transmission windows to be advanced while others are delayed, causing them to collide. Moreover, given the small variability of the clocks, the resulting hyper-periods would be exponentially longer. For example, if two legacy systems transmit with 1 s period, but one has a 1% positive drift and the other one has 1% negative, instead of a GCL of 1 s with three transmission windows, the GCL would have an extension of lcm (1.01,0.99)=99.99 s with more than 200 transmission windows. Fig. 14 illustrates an example of a heterogeneous TSN network following a tree topology combining two legacy systems with different drifts. The TSN communication subnetwork comprises switches SW1, SW2, and SW3, while legacy systems 1 and 2 consist of end-stations ES1.1 and ES1.2, and ES2.1 and ES2.2, respectively. According to the requirement, TT traffic routes of each legacy system do not share output ports with the TT traffic routes of the other legacy system. To ensure meeting this condition, the TT traffic of each legacy system is grouped into separate branches of the tree topology, and the intersystem communication is restricted to AVB or BE traffic.

VI. TALESS VALIDATION SETUP

In this article, we validate the solution's effectiveness using two methods: a simulation model of the solution at the end-stations and an experimental implementation.

A. SIMULATION MODEL

Our model simulates the behavior of a TSN switch implementing TALESS. However, since TALESS solely eliminates drift, the results obtained in our experiments can be extrapolated to more extensive networks with any type of schedule, as long as the network architecture and schedule are functional in the absence of drift. The model is implemented in MATLAB and uses several parameters as inputs. These parameters include the period of the transmission to be modeled, the drift at the end of the experiment, and the jitter of the received transmission as the variance of a specified distribution. In addition, the modeled network run-time must be specified as an input. This is one of the main advantages of the model over the experimental implementation since, as real drifts are very small, the effects are noticeable only in the long term. In this sense, the model allows us to analyze long periods of time with realistic drift values in a reasonable model execution time.

The reception of frames is modeled as a list of timestamps (ts) generated by applying the drift variation (dv) and jitter (j) to the period (p), i.e.

$$ts_i = ts_{i-1} + p \times dv^i + \text{normrnd}(0, var) \tag{2}$$

where normrnd(0, var) is a random value following a specific distribution, in this case, a normal distribution, with mean 0 and the variance var corresponding to the variance of the jitter used as an input. The DD module analyzes all ts values in the list individually. The DD module determines whether the period of the reception is equal to the initially scheduled one using a Student's t-test (`ttest`[2]). Once a significant difference is detected, i.e., the probability of the periods being equal is below a predetermined threshold, the period is updated based on the trend measured in the frames received since the last period update (`polyfit`[3]).

Finally, the model shows three different results for both positive and negative drift. The first result is the behavior of the reception with free transmission, i.e., without the intervention of the TSN switch, while the second result is the behavior with a fixed schedule without applying any solution. Finally, the last result is the effect of TALESS implementation. The results will be presented and discussed in Section VII.

B. EXPERIMENTAL SETUP

We extended the network presented in Section IV for the experimental implementation. More specifically, we use 4 Raspberry PIs and 4 TSN switches, and a computer that will

[2]One-sample and paired-sample t-test - MATLAB ttest [Online]. Available: https://se.mathworks.com/help/stats/ttest.html

[3]Polynomial curve fitting - MATLAB polyfit - MathWorks [Online]. Available: https://se.mathworks.com/help/matlab/ref/polyfit.html

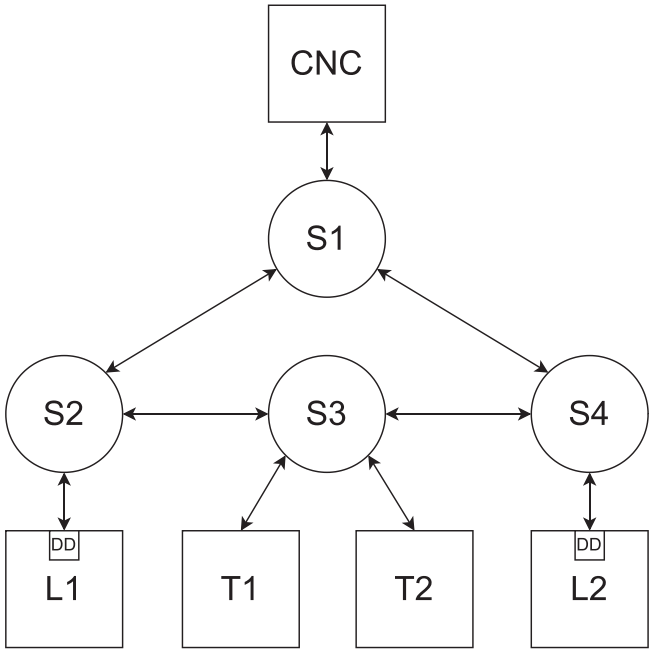

FIGURE 15. Experimental network diagram showing TSN Switches (S) and legacy systems 1 and 2 represented by Talkers (T) and Listeners (L).

act as a CNC. The architecture of the new network is illustrated in Fig. 15, where T1 and L1 represent the talker and listener of legacy system 1, and T2 and L2 the ones of legacy system 2. In addition, S1 to S4 and the CNC represent the TSN communications subsystem.

Each pair of Raspberry PIs (Ti, Li) forms an independent legacy system, i.e., they are not synchronized nor communicate with the end-stations of the other legacy system. For each talker, we implemented a synthetic clock with different drift values with respect to the TSN communications subsystem that changes throughout the experiment. This synthetic clock only changes the time perception of the legacy system by applying certain drift to the local clock synchronized through NTP. For example, if a 10% drift is applied, the synthetic clock will multiply all times by 1.1. These drift values were larger than those present in a normal network to magnify the effects in a reasonable duration of the experiments. In addition, the drift grew positively in one of the legacy systems, while in the other, it grew negatively. Each legacy system's synthetic clock is responsible for driving the transmission. To keep the talker and the listener synchronized with the drift changes, apart from the previously mentioned NTP, every time the synthetic clock drift changes, the talker sends a message to the listener with the new drift value so that the listener can update its synthetic clock.

According to the design sketched in Section V, the DD should be implemented in the input port of switches to avoid modifications in the legacy end-stations. However, since we do not have access to the implementation of switches, we implemented the DDs in the legacy listener. Despite the change of the DDs location, neither the calculation method nor the obtained drift value changes. This is because the DDs can monitor the drifts on the ports, either connected to switches or the legacy end-stations. Once the DD measures a significant clock difference, it sends the drift value to the CNC. Note that the addition of the DD is the only modification made to the legacy end-stations with respect to their original implementation. This is required due to the limitations in modifying the commercial TSN switches. However, in a real TALESS implementation, no modifications to the legacy end-stations would be necessary.

The DD samples the frame reception time, either at the legacy end-station or at the TSN switch port connected to one, and compares it with the scheduled reception period. Using a t-test, it analyzes if there are variations in the periodicity. If so, the DD calculates the drift by dividing the period measured by the scheduled one and sends it to the CNC.

The CNC is based on the implementation proposed in [31], which was openly available to the research community. It uses a JSON file with the configuration to be deployed in the TSN network and NETCONF to deploy the configuration. The CNC is implemented to receive drift information from the DD, update the configuration based on (1), and automatically deploy the improved configuration in the TSN network. The results are presented and discussed in Section VII.

VII. SIMULATION AND EXPERIMENTAL RESULTS

This section will show and analyze the results obtained using the simulation model and TALESS experimental implementation. In addition, we will compare the model with the experimental implementation to verify both the implementation and the simulation model.

We will use a metric called Synchronization Quality Metric (SQM) to analyze the obtained results. This is calculated by dividing the difference between the Reception Time (RT) of two consecutive frames minus the Scheduled Period (SP) for those frames by the SP, i.e.,

$$SQM_i = \frac{(RT_{i+1} - RT_i) - SP}{SP}. \tag{3}$$

The SQM allows us to analyze drift and jitter graphically. On the one hand, the mean SQM in a given interval provides information about the drift. Since the SQM gives the variation between the reception period and the scheduled period, if, for example, the scheduled period is 1 time unit and the average SQM is 0.1, then the system is receiving with a period of 1.1 time units. Therefore, there is a drift of 10%, i.e., the frames will arrive at times 1.1, 2.2, and 3.3 when they should arrive at times 1, 2, and 3. On the other hand, the maximum absolute value of SQM minus the mean SQM provides the ratio of jitter with respect to the period since by eliminating the drift from the variations in reception, we obtain the variation caused by the jitter. Note that this metric does not allow us to observe extreme cases such as frame loss, since the SQM cannot be quantified due to the missing RT.

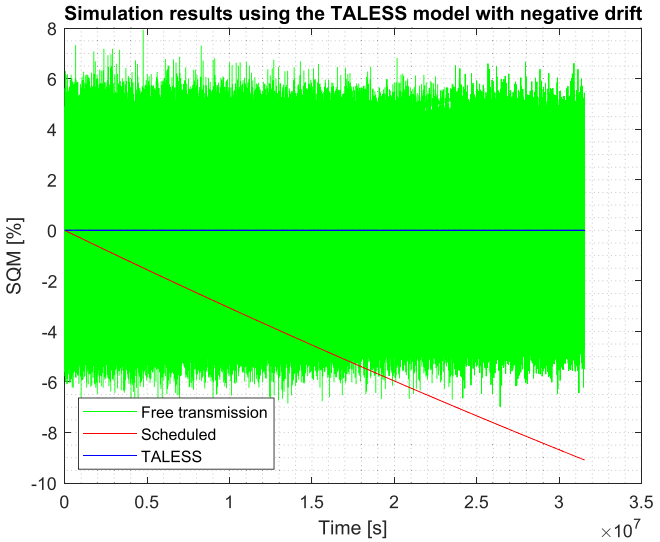

FIGURE 16. Simulation results of one year of transmissions in a heterogeneous TSN network with negative clock drift in three different scenarios: free, scheduled, and TALESS transmission.

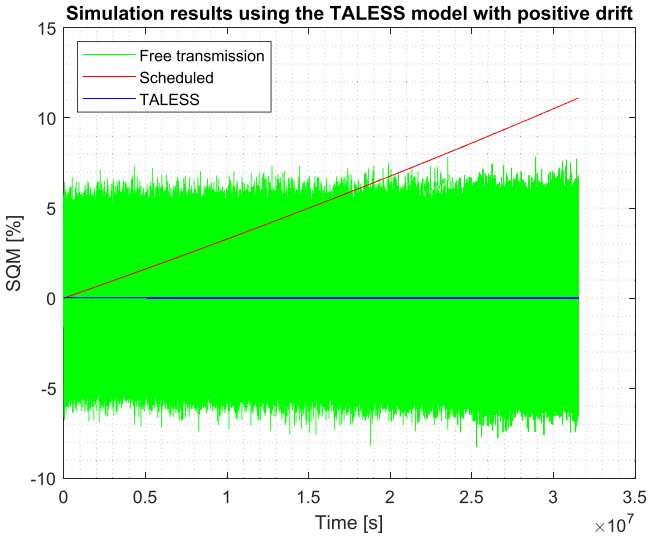

FIGURE 17. Simulation results of one year of transmissions in a heterogeneous TSN network with positive clock drift in three different scenarios: free, scheduled, and TALESS transmission.

Finally, all the analyses will be performed by comparing the reception of periodic frames in three different scenarios: (i) with free traffic flow through the TSN network, i.e., without applying TAS or any other scheduling mechanism, (ii) with the TSN communications subsystem scheduled without TALESS implementation, and (iii) with TALESS implementation.

A. SIMULATION MODEL RESULTS

We simulate two different scenarios using the simulation model. In both cases, the model simulates a year of communications of a periodic transmission with an initial period of 1 s and with a variable drift that starts at 0% and grows progressively until reaching 10% at the end of the simulation in the first scenario and from 0 to -10% in the second one. These drift variations reflect the natural degradation of the end-station clocks, either positive or negative, caused by factors such as the passage of time or environmental influences like temperature, pressure, or electromagnetic interference. In addition, the jitter of the transmission is used as an input to the model and follows a normal distribution of variance 0.01. This distribution and variance are similar to the ones in Section IV. The results can be seen in Figs. 16 and 17, both showing the SQM over the simulation time.

In both scenarios, the free reception has a jitter of 70 ms (as defined as input) and zero drift. When scheduling the TSN subsystem without TALESS, the jitter almost disappears, but the effects of the drift between the TSN schedule and the legacy transmission become evident. Finally, we observe that by applying TALESS, both the jitter and the drift almost drop to 0.

B. REAL NETWORK IMPLEMENTATION RESULTS

Using the real network, we run an experiment similar to the model but with certain restrictions. Instead of a year of execution, only 2000 frames are transmitted in each legacy system, and the drift, instead of increasing and decreasing progressively up to $\pm 10\%$, varies by $\pm 5\%$ every 100 frames. Moreover, the legacy system with positive drift starts with a period of 1 s that is periodically shortened, while in the negative drift legacy system, it is the final period, which is equal to 1 s. All other characteristics are the same as in the simulation model. This experiment covers all the scenarios considered in this study, including the lack of synchronization between TSN and the legacy systems, the presence of drift due to the lack of synchronization, a time-varying drift due to environmental conditions (temperature, vibrations, and power-supply, etc.), and the coexistence of two legacy systems with distinct drift characteristics. The results of these experiments can be seen in Figs. 18 and 19.

As in the model, we can see how the free transmission has high jitter and no drift. Once the scheduling is applied without TALESS, the jitter disappears, but the drift occurs. In this case, the SQM (and therefore the drift) presents a step-wise behavior instead of a continuous one because, as previously mentioned, the drift variation is applied every 100 frames for simplicity in the experiment.

Finally, we see how TALESS eliminates jitter and drift yet leaves some drift remnants (the duty cycle observed in the figures). These are due to the time required by the solution to detect the change in the reception and are larger than what is observed in the simulation model due to the large synthetic drift applied to this experiment to allow us to visualize the effects of TALESS on the drift in a reasonable time. Although

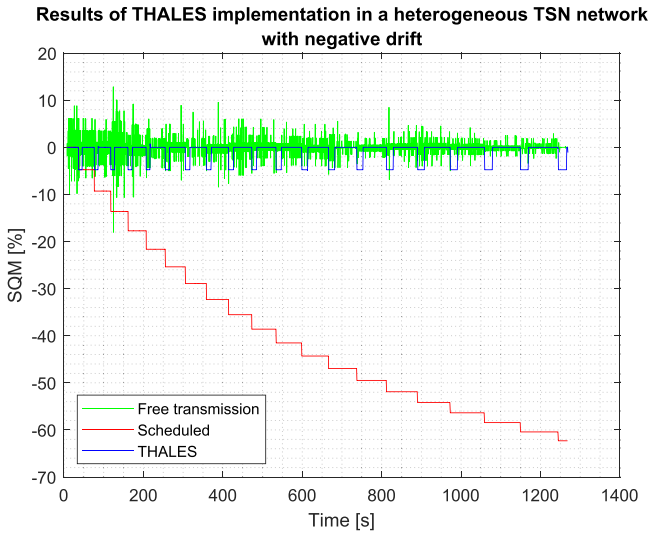

FIGURE 18. Results of heterogeneous TSN network execution with negative drift in three different scenarios: free, scheduled, and TALESS transmission.

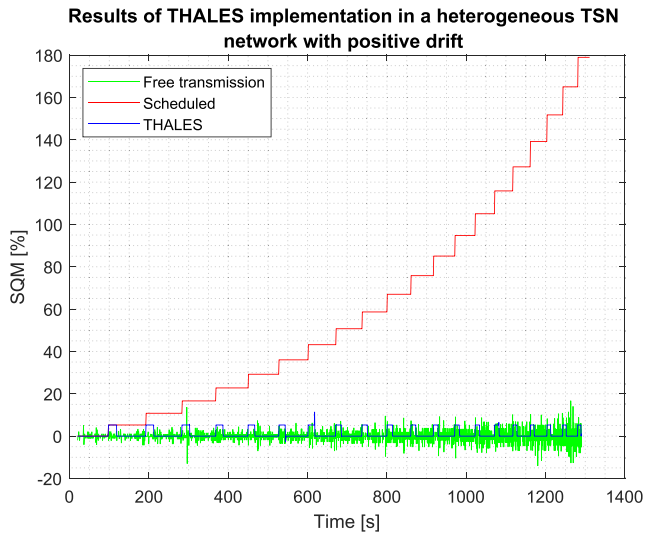

FIGURE 19. Results of heterogeneous TSN network execution with positive drift in three different scenarios: free, scheduled, and TALESS transmission.

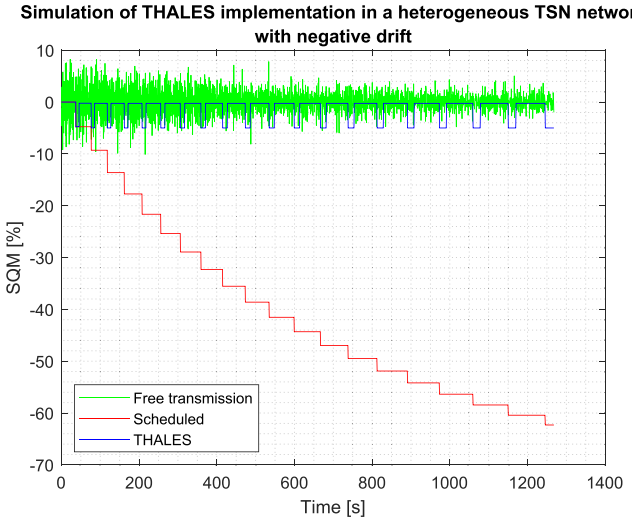

FIGURE 20. Simulation results of the implemented heterogeneous TSN network with negative drift.

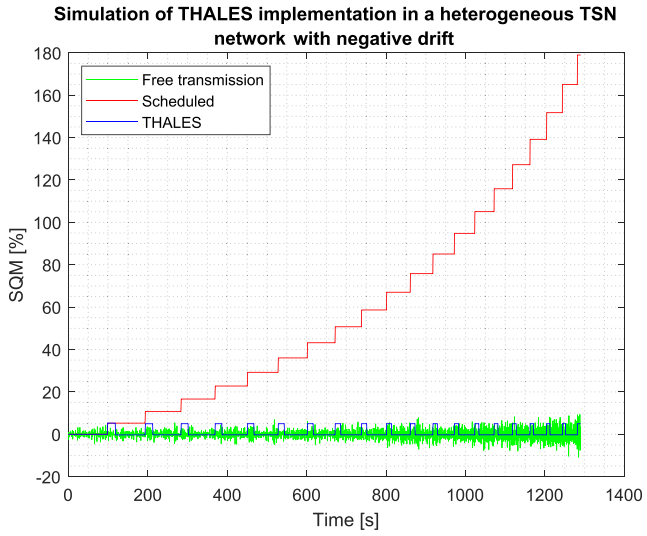

FIGURE 21. Simulation results of the implemented heterogeneous TSN network with positive drift.

small periodic drifts can accumulate significant clock skew between the TSN network and the legacy system, there are ways to prevent this, e.g., by over-correcting the drift by creating equivalent drifts but of opposite sign to ensure an overall average drift equal to 0.

Also, as described in Section VI, in this experiment, both the positive and negative drift scenarios are performed simultaneously in the same network. This demonstrates that TALESS is capable of handling different drifts simultaneously. However, the difference in drift between legacy systems makes communication impossible through TT traffic. This can be achieved through other types of traffic not sensitive to clock drift, such as AVB or BE, improving the integrability of the different legacy systems integrated into TSN.

C. COMPARISON RESULTS

Finally, to verify both the simulation model and the experimental implementation, we modified the model to simulate with the same conditions applied to the experimental implementation, i.e., execution of only 2000 frames with a variable drift of ±5% every 100 frames. Such simulations' results are shown in Figs. 20 and 21.

As we can see, the simulations of implemented scenarios match the implementation results. Although the transmission by the legacy talker is not exactly the same since the real network does not strictly follow a normal distribution, the

TABLE 1 Results in Absolute Values

Experiment		Jitter	Traffic Skew
Simulation	Free Transmission	70 ms	0 ms
	Scheduled	0 ms	18 h
	TALESS	2 ms	0 ms
Real Network	Free Transmission	70 ms	0 ms
	Scheduled	≈ 0 ms	1300 s
	TALESS	1 s	≈ 0 ms

effects of both schedulings (with and without TALESS) on reception are essentially the same. This experiment provides evidence that the simulation model follows the experimental results, ensuring the validity of the simulation model and, therefore, of TALESS.

In Table 1, we outline the absolute values of the jitter and drift obtained from both the simulation and the real network. Regarding drift, we calculate the difference between transmission and schedule time solely at the end of the experiments. At the start, the clock skew is presumed zero as insufficient time has elapsed for clock divergence. Note that the simulation spans one year, while the real network experiment lasts 2000 s of clock time for the end-station, under artificially amplified drift. In both simulated and real network experiments, the free transmission showcases the results previously discussed. Regarding the scheduled transmission without TALESS, both the real network and the simulation showed zero jitter. In the simulation, this occurs because we do not model TSN jitter since TSN time is directly equated with real-time, while real network results lack precision for direct jitter measurement, though TSN specifications suggest nanosecond-scale jitter. Notably, discrepancies between the end-station clock time and the TSN schedule due to clock drift are evident in both scenarios at the end of the experiments. Introducing TALESS effectively eliminates drift, yet simulation indicates minor jitter due to the time required for drift detection and application, while real network results show a 1 s jitter, attributed to the artificially amplified drift variation applied in the experiments.

D. RECONFIGURATION TIME

Determining the network reconfiguration time is crucial for assessing the achievable jitter in the network. This time encompasses the duration needed to detect and address drift when it arises. By calculating this time, we can ascertain the traffic skew achieved before implementing the solution. This traffic skew, combined with the TSN jitter, determines the jitter experienced by the legacy system traffic when the legacy system experiences a drift change. This parameter influences scheduling factors such as transmission window offsets. If the transmission window offset is bigger than the maximum drift plus the transmission jitter of the legacy end-station, we ensure that, in the absence of any other issues, TT frames will consistently transmit within their designated windows. Consequently, it becomes feasible to define latency by design by scheduling the TT traffic transmission windows via the TAS's GCL. Moreover, the reception jitter will be zero, while the latency jitter will be constrained to the transmission jitter of the legacy end-station plus the maximum traffic skew, which represents less than 2% of the jitter in the experiments as we will see as follows.

The reconfiguration time can be dissected into three components: drift detection time, rescheduling time, and new schedule deployment time. For drift detection, in our experimental network with approximately 70 ms jitter and around 1 μs drift per second, the DD requires sampling 2000 frames to detect the drift. Lower jitter and higher drift necessitate fewer samples. In our scenario, the 2000 frames needed imply a maximum traffic skew of 0.2% of the sampled frame period, corresponding to 2 ms for frames with a 1-s period.

As for rescheduling time, it can be considered negligible since TALESS modifies the existing schedule using (1) rather than creating a new one.

Regarding new schedule deployment time, the CNC enables background preparation of the new schedule, allowing the application at an opportune moment, such as the end of a GCL cycle. Thus, in the worst case, one clock cycle would be added to the network reconfiguration time. Given our legacy clock parameters, this translates to an additional traffic skew of 0.0001% of the GCL cycle.

In summary, the reconfiguration time equals the duration needed for drift identification plus one GCL cycle. For our experimental legacy network model, this time amounts to 2000 periods of the sampled frame plus one GCL cycle, totaling 2001 s. This corresponds to a maximum traffic skew of approximately 2 ms before implementing the solution.

VIII. CONCLUSION

This article analyzed the effects of the lack of synchronization between the legacy systems and the TSN network. These effects are mainly due to the drift between TSN clocks and legacy systems, resulting in either delayed TT transmission or missing frames in the long term. Therefore, we designed, implemented, and validated a solution, TALESS, to remove the identified effects. Through simulation and implementation of TALESS, we demonstrated that TALESS efficiently enforces the reduction of jitter and removes the effects of clock drifts in legacy systems. This solution allows us to integrate several legacy systems into a TSN network without modifying their clock synchronization.

In future work, we aim to implement the proposed mechanism within a TSN switch to provide a complete tool for TSN adoption without any modification within the legacy systems.

REFERENCES

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017, doi: 10.1109/MIE.2017.2649104.

[2] S. Samii and H. Zinner, "Level 5 by layer 2: Time-sensitive networking for autonomous vehicles," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 62–68, Jun. 2018, doi: 10.1109/MCOMSTD.2018.1700079.

[3] R. Salazar, T. Godfrey, N. Finn, C. Powell, B. Rolfe, and M. Seewald, "Utility applications of time sensitive networking white paper," *Utility Appl. Time Sensitive Netw. White Paper*. Piscataway, NJ, USA, pp. 1–19, 2019.

[4] M. Bajer, "Dataflow in modern industrial automation systems. theory and practice," *Int. J. Appl. Control Electr. Electron. Eng*, vol. 2, no. 4, pp. 1–11, 2014.
[5] J. Feld, "PROFINET-scalable factory communication for all applications," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, 2004, pp. 33–38.
[6] D. Jansen and H. Buttner, "Real-time ethernet: The EtherCAT solution," *Comput. Control Eng.*, vol. 15, no. 1, pp. 16–21, 2004.
[7] SERCOS International E.V., "SERCOS - the real-time ethernet communication standard for motion control," [Online]. Available: https://www.sercos.org, 2024, accessed: 2024-04-15.
[8] *IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications, IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020, doi: 10.1109/IEEESTD.2020.9121845.
[9] D. Bujosa, A. Johansson, M. Ashjaei, A. V. Papadopoulos, J. Proenza, and T. Nolte, "The effects of clock synchronization in TSN networks with legacy end-stations," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Automat.*, 2022, pp. 1–4.
[10] C. Latha and H. Shashidhara, "Clock synchronization in distributed systems," in *Proc. 5th Int. Conf. Ind. Inf. Syst.*, 2010, pp. 475–480.
[11] J. Haxhibeqiri, X. Jiao, M. Aslam, I. Moerman, and J. Hoebeke, "Enabling TSN over IEEE 802.11: Low-overhead time synchronization for wi-fi clients," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2021, vol. 1, pp. 1068–1073.
[12] H. Baniabdelghany et al., "Extended synchronization protocol based on IEEE802. 1AS for improved precision in dynamic and asymmetric TSN hybrid networks," in *Proc. IEEE Mediterranean Conf. Embedded Comput.*, 2020, pp. 1–8 .
[13] A. M. Romanov, F. Gringoli, and A. Sikora, "A precise synchronization method for future wireless TSN networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3682–3692, May 2021.
[14] O. Seijo, X. Iturbe, and I. Val, "Tackling the challenges of the integration of wired and wireless TSN with a technology proof-of-concept," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7361–7372, Oct. 2022.
[15] M. Gundall, C. Huber, P. Rost, R. Halfmann, and H. D. Schotten, "Integration of 5 G with TSN as prerequisite for a highly flexible future industrial automation: Time synchronization based on IEEE 802.1 AS," in *Proc. IECON Annu. Conf. IEEE Ind. Electron. Soc.*, 2020, pp. 3823–3830.
[16] Z. Chai, W. Liu, M. Li, and J. Lei, "Cross domain clock synchronization based on data packet relay in 5G-TSN integrated network," in *Proc. IEEE Int. Conf. Electron. Commun. Eng.*, 2021, pp. 141–145 .
[17] J. Song, M. Kubomi, J. Zhao, and D. Takita, "Time synchronization performance analysis considering the frequency offset inside 5G-TSN network," in *Proc. IEEE Int. Symp. Wireless Commun. Syst.*, 2021, pp. 1–6 .
[18] H. Shi, A. Aijaz, and N. Jiang, "Evaluating the performance of over-the-air time synchronization for 5 G and TSN integration," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw.*, 2021, pp. 1–6 .
[19] T. Striffler and H. D. Schotten, "The 5 G transparent clock: Synchronization errors in integrated 5G-TSN industrial networks," in *Proc. IEEE Int. Conf. Ind. Informat.*, 2021, pp. 1–6 .
[20] J. Xue, G. Shou, H. Li, and Y. Liu, "Enabling deterministic communications for end-to-end connectivity with software-defined time-sensitive networking," *IEEE Netw.*, vol. 36, no. 2, pp. 34–40, Mar./Apr. 2022.
[21] D. Bujosa, D. Hallmans, M. Ashjaei, A. V. Papadopoulos, J. Proenza, and T. Nolte, "Clock synchronization in integrated tsn-ethercat networks," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Automat.*,2020, vol. 1, pp. 214–221 .
[22] S. Szancer, P. Meyer, and F. Korf, "Migration from SERCOS III to TSN-Simulation based comparison of TDMA and CBS transportation," in *Proc. OMNeT*, 2018, pp. 52–62 .
[23] S. Nsaibi, L. Leurs, and H. D. Schotten, "Formal and simulation-based timing analysis of industrial-ethernet sercos III over TSN," in *Proc. IEEE/ACM 21st Int. Symp. Distrib. Simul. Real Time Appl.*, 2017, pp. 1–8.
[24] S. Schriegel and J. Jasperneite, "A migration strategy for profinet toward ethernet TSN-based field-level communication: An approach to accelerate the adoption of converged IT/OT communication," *IEEE Ind. Electron. Mag.*, vol. 15, no. 4, pp. 43–53, Dec. 2021.
[25] M. Barzegaran, N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Real-time traffic guarantees in heterogeneous time-sensitive networks," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, 2022, pp. 46–57 .
[26] Z. Zhao et al., "Access mechanism for period flows of non-deterministic end systems for time-sensitive networks," *Comput. Netw.*, vol. 231, 2023, Art. no. 109805.
[27] D. L. Mills, "Network time protocol (NTP)," Tech. Rep., 1985.
[28] P. Pedreiras, L. Almeida, and P. Gai, "The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency," in *Proc. 14th Euromicro Conf. Real-Time Syst.* 2002, pp. 152–152.
[29] R. Enns, "RFC 4741: NETCONF Configuration Protocol," 2006.
[30] *IEEE 802.1Qcc-2018 - IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 17: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, 2018, doi: 10.1109/IEEESTD.2018.8372875.
[31] I. Alvarez, A. Servera, J. Proenza, M. Ashjaei, and S. Mubeen, "Implementing a first CNC for scheduling and configuring TSN networks," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Automat.*, 2022, pp. 1–4 .

DANIEL BUJOSA MATEU (Student Member, IEEE) born in Palma de Mallorca, Spain, in 1995. He received a bachelor's degree in automation and industrial electronic engineering and the master's degree in industrial engineering from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 2017 and 2019, respectively, and the Licenciate thesis degree from the Mälardalen University (MDU), Västerås, Sweden, in 2023. He is currently working towards the Ph.D. degree with the School of Innovation, Design, and Engineering, Mälardalen University.

His research interests include distributed embedded systems, real-time communications, and legacy systems support.

JULIAN PROENZA (Senior Member, IEEE) received the first degree in physics (Licenciatura en Ciencias Físicas) and the doctorate degree in informatics from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 1989 and 2007, respectively.

He is currently a Lecturer with the Department of Mathematics and Computer Science, UIB. His current research interests include dependable and real-time systems, fault-tolerant distributed systems, adaptive systems, clock synchronization, dependable communication topologies, and field-bus and industrial networks.

Dr. Proenza has been a member of the IEEE Industrial Electronics Society (IES) since 2009 and a Senior Member since 2012. He is also a member of the IES Technical Committee on Factory Automation.

ALESSANDRO V. PAPADOPOULOS (Senior Member, IEEE) received the B.Sc. and M.Sc. (*summa cum laude*) degrees in computer engineering from the Politecnico di Milano, Milan, Italy, in 2008 and 2010, respectively, and the Ph.D. (Hons.) degree in information technology from the Politecnico di Milano, in 2013.

He is currently a Full Professor of electrical and computer engineering with Mälardalen University, Västerås, Sweden, and a QUALIFICA Fellow with the University of Málaga, Spain. Since March 2024, he has been the Scientific Leader of Applied AI with Mälardalen University. He was a Postdoctoral Researcher with the Department of Automatic Control, Lund, Sweden (2014–2016) and Politecnico di Milano, Milan, Italy (2016). His research interests include robotics, control theory, real-time systems, and autonomic computing.

Dr. Papadopoulos was the Program Chair for the Mediterranean Control Conference (MED) 2022 and the Euromicro Conference on Real-Time Systems (ECRTS) 2023. He is an Associate Editor for the *ACM Transactions on Autonomous and Adaptive Systems, Control Engineering Practice, and Leibniz Transactions on Embedded Systems*.

THOMAS NOLTE (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Mälardalen University, Västerås, Sweden, in 2006.

He has been a Visiting Researcher with the University of California, Irvine (UCI), Los Angeles, CA, USA, in 2002, and a Visiting Researcher with University of Catania, Catania, Italy, in 2005. He has been a Postdoctoral Researcher with the University of Catania, in 2006, and with MDU from 2006 to 2007. He has been a Full Professor of Computer Science with MDU, since 2012. He has been the Director of the industrial PhD school Automation Region Research Academy (ARRAY), since 2017, and Research Leader of research in Electrical and Computer Engineering with MDU, since 2022. He has been the Scientific Advisor with ABB since 2012 (2012-2016 @ ABB Corporate Research, and since 2017 @ ABB Robotics). He has co-authored more than 350 scientific papers.

MOHAMMAD ASHJAEI (Senior Member, IEEE) received the Ph.D. degree in computer science from Mälardalen University (MDU), Vesteras, Sweden, in 2016.

He is currently an Associate Professor with the Complex Real-Time Systems (CORE) and the Heterogeneous Systems – Hardware Software Co-design (HERO) research groups at MDU, Sweden. He is also giving lectures on various topics related to embedded systems and data communication networks. His main research interests include real-time systems, real-time distributed systems, scheduling algorithms on networks and processors, schedulability analysis techniques, resource reservation, and reconfiguration mechanisms for real-time networks.

Dr. Ashjaei is a PC member and referee for several international conferences and journals, including IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII), IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS (TIE), *Elsevier's Journal of Systems Architecture*, IEEE TRANSACTIONS ON NETWORK AND SERVICES, *Journal of Cloud Computing, and ACM Computing Surveys*. He has organized and chaired several tracks, conferences, special sessions and workshops at international conferences.