IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 July 2026, accepted 31 July 2026, date of publication 5 August 2026, date of current version 11 August 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3720992

RESEARCH ARTICLE

A Digital Twin-Enabled Management Framework for Low-Latency and Low-Jitter Wireless TSN

PABLO AVILA-CAMPOS, (Graduate Student Member, IEEE), JETMIR HAXHIBEQIRI, (Senior Member, IEEE), XIANJUN JIAO, INGRID MOERMAN, (Fellow, IEEE), AND JEROEN HOEBEKE
IDLab, Department of Information Technology, IMEC—Ghent University, 9052 Ghent, Belgium

Corresponding author: Pablo Avila-Campos (pabloesteban.avilacampos@ugent.be)

This work was supported in part by the IMEC ICON Project VELOCe—Verifiable, Low-Latency Audio Communication (Agentschap Innoveren en Ondernemen Project) under Grant nr.HBC.2021.0657; in part by the Flemish Fonds Wetenschappelijk Onderzoek (FWO) Strategic Basic Research (SBO) VERI-END.com Project under Grant S003921N; in part by the UNITY-6G Project, funded by European Union's Horizon Europe SNS JU Program under Agreement 101192650; and in part by the Flemish Government under the "Onderzoek-Sprogramma Artificiele Intelligentie (AI) Vlaanderen" Program.

ABSTRACT Time-Sensitive Networking (TSN) provides guaranteed traffic delivery, making it essential for industrial automation, multimedia, automotive systems, and other areas. Although TSN is well-established in wired networks, wireless systems face extra challenges such as delays, interference, and unstable links. Extending TSN to wireless (WTSN) thus adds complexity, particularly in managing traffic. This paper proposes a data-driven solution leveraging Digital Twin (DT) modeling to enhance WTSN management. We focus on mitigating the residual service time (RST) problem introduced by non-TSN traffic generators, which increases link latency. Implemented in a real Wi-Fi TSN-based environment, our implementation demonstrates that delay prediction through WTSN-DT reduces application link latency by up to 96% at the 90th percentile in a real Wi-Fi TSN setting, relative to fixed time slot allocation.

INDEX TERMS Wireless time-sensitive networking, Wi-Fi, ML, residual service time.

I. INTRODUCTION

From industrial automation, multimedia, and automotive to Mars exploration, time-sensitive networking (TSN) is proposed as a solution to ensure traffic delivery guarantees and network determinism [1], [2], [3]. Such guarantees are achieved through a combination of methods focused on time synchronization [4], low latency [5], reliability [6], and resource management [7]. Although initially designed for wired networks, these guarantees are particularly beneficial for wireless networks. Wireless systems often face delays, and interference is pervasive due to the shared and variable medium, creating unreliable links. Moreover, wireless networks typically operate in half-duplex mode, resulting in lower capacity compared to full-duplex wired networks [8].

Extending TSN to wireless communication (WTSN) enhances deterministic networks with flexibility and mobility, making it suitable for locations where wired networks are impractical. However, this extension introduces significant challenges, particularly in traffic management [9]. While practical TSN management solutions are already limited in wired contexts, addressing them in WTSN is even more difficult due to the unpredictability of wireless communication.

Traffic management aims to balance traffic characteristics and requirements, available resources, and the inherent variability of wireless networks. Accurate WTSN management must also account for factors like variable delays in frame processing, channel propagation, transmission, time synchronization accuracy, and frame generation in heterogeneous hardware [10]. Ignoring these factors reduces determinism and degrades timing guarantees required by time-sensitive applications [11].

A key reason for this lack of realism is the absence of a management framework that accurately models network complexities without oversimplifying, as traditional

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Fadda.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

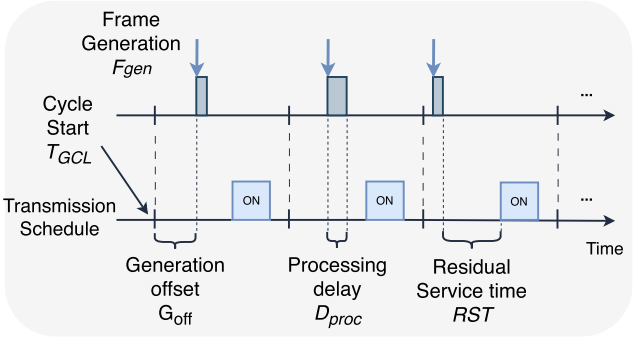


FIGURE 1. RST challenge in a TSN node.

approaches often do [12]. In this work, we propose a management framework to bridge this gap. To demonstrate its potential, we focus on the residual service time (RST) problem, as illustrated in Figure 1. The figure highlights how mismatches between cyclic frame generation (F_{gen}) relative to the cycle start (T_{GCL}) of the transmission schedule based on the TSN Time-Aware Shaper (TAS) [5] and variable frame processing delays (D_{proc}) at TSN nodes affect the frame time slot Residual Service Time (RST). These mismatches, often caused by non-TSN traffic generators, increase end-to-end latency and its variability, leading to jitter [13].

Network jitter refers to the variation in latency or delay of packet delivery over a network. Jitter challenges TSN core goal of achieving predictable, low-latency communication. If not addressed, it can block critical traffic from accessing its reserved time slot, cause collisions, or introduce delays, thereby compromising the system's determinism. Therefore, minimizing jitter is not merely beneficial but essential for TSN to operate as intended [14].

Current traffic scheduling demonstrations often use TAS with short Gate Control List (GCL) cycle times. These short cycles produce a small RST, which effectively masks underlying scheduling issues. A numerical example appears in Section IV-A. Alternatively, the strategy proposed in 3GPP Release 18 holds frames in the buffer until they reach a defined maximum delay. While this reduces jitter, it increases end-to-end latency, limiting its usability in time-sensitive applications [15], [16].

To address these limitations, a data-driven real-time management framework is required. The Digital Twin (DT) concept has emerged as a promising solution for addressing this challenge [17], [18]. The IETF and ITU-T recognize Network-DT as a key technology for achieving closed-loop network automation, proposing a reference architecture that fully integrates DT within a general network controller management framework. However, these efforts still lack specific guidance for TSN [19], [20].

Unlike a standard centralized controller, the DT maintains a real-time, predictive model of each WTSN node and their interactions, enabling real-time scheduling adjustments that, in this specific case, is focused on mitigating the RST caused by heterogeneous and non-time-synchronized traffic, but may be extended to other scenarios, a capability beyond the reach of conventional controllers.

In this work, we present an initial exploration of the feasibility of combining a Wi-Fi-based (IEEE 802.11n) WTSN with a DT, a solution we call WTSN-DT, to improve traffic management through scheduling. While the current latest Wi-Fi standard (IEEE 802.11be) lacks a native TAS, we leverage the custom gating system of the openwifi[1] platform to implement deterministic traffic management [21]. This allows us to address the critical challenge of minimizing wireless communication link jitter and end-to-end latency introduced by RST. This challenge becomes especially significant in the presence of heterogeneous nodes and non-time-synchronized traffic sources. Because non-TSN end nodes are likely to be among the first to benefit from WTSN, bridging this gap is essential. Accordingly, the main contributions of this work are as follows:

- Propose an initial Wi-Fi-based WTSN-DT management reference framework that integrates real-time measurements, modeling, scheduling, and setting.
- Investigate three analytical and Machine Learning (ML)-based modeling strategies for real-world cyclic traffic patterns based on frame arrival time series.
- Implement and test a WTSN-DT element aimed at reducing link latency by scheduling transmission time slots in a real Wi-Fi-based WTSN setup using openwifi.

The remainder of the paper is organized as follows. Section II presents related work on DT and WTSN. Section III describes an initial proposed integration of TSN control and DT. Section IV explains the proposed analytical and AI-based predictive traffic modeling strategies. Section V presents the setup and results from both the simulated and practical implementations. Finally, Section VI concludes the paper.

II. RELATED WORKS

Interest in introducing the DT concept into networking is not new. The Networking Digital Twin has been proposed as a powerful tool to connect physical and virtual digital spaces. According to ITU-T, it enables real-time management of complex networks by using physical models to replicate their states, providing high fidelity and seamless integration [19], [20], [22].

An example of this is found in [18], where the authors propose a DT data pipeline architecture that demonstrates an integrated structure for network modeling flows. They also introduce a feature extraction technique for traffic modeling to optimize learning time. Although not focused on TSN, this work highlights gaps in data flow integration when combining networking, digital twins, and time series modeling. These gaps are even more pronounced in the TSN environment.

Efforts to use DT for specific issues, such as network jitter, have been explored in works like [17]. This study highlights

[1] https://github.com/open-sdr/openwifi

the need for solutions in wireless networks, particularly those aligned with 3GPP standards. It proposes training a deep learning agent to learn inter-packet arrival patterns and control buffer times to reduce jitter. While the approach effectively minimizes jitter, it increases end-to-end latency, making it less suitable for TSN, where bounded low latency is a critical requirement.

From a different perspective, [23] showcases the advantages of TSN as the transport layer for Digital Twin-based manufacturing systems. The deterministic latency guarantees of TSN are essential for high-fidelity virtual replication, directly enhancing prediction accuracy for dynamic systems, a key requirement in your DT-based TSN management approach.

Another challenge in integrating DT with wireless networks is reliability. In the context of Wi-Fi 7 multi-link operation (MLO), the authors of [24] propose WiTwin, a centralized DT designed to enable seamless handovers. Handovers are essential for time-sensitive applications, as they must support mobility while maintaining performance guarantees. However, while the study addresses time-sensitivity, it does not focus on TSN-specific control elements or their interactions.

Closer to the current work, [25] presents a broader vision for an end-to-end control plane supporting multi-domain and multi-technology communication systems, including Wi-Fi, with their federated DT model sharing similar goals to this work for the Wi-Fi case but still far from practical implementation. Similarly, other proposals explore integrating TSN and DT with future technologies like 6G, such as [26], which identifies DT as a core component for integrating TSN with 6G. While their theoretical approach addresses challenges like network complexity, infrastructure, and data collection, both approaches share the common issue of lacking a TSN DT data model, which is crucial for real-world adoption.

Regarding the specific issue of RST, existing research remains limited. Current WTSN implementations typically manage a small number of traffic flows with similar or identical generation cycles and often lack dedicated time slots for time synchronization or network management. When the GCL cycle is short, jitter exists but is less noticeable. However, as GCL cycle durations increase, as described in the Introduction, jitter becomes a significant problem [27].

Close to the present work, in [28], the authors address TSN jittering from a missynchronization perspective in non-TSN end devices by introducing TALESS, a solution that compensates for drift between TSN and non-TSN devices by adjusting the transmission schedule. A key feature of their approach is a drift detector (DD) that continuously monitors synchronization drift at the reception port and calculates the drift percentage.

Our approach differs in several key aspects. First, we do not consider frame losses, as synchronization drift does not cause frame loss when the generation and GCL cycles are similar. Second, our focus on wireless communication introduces a different topology. While our capture mechanism is decentralized, it incorporates both generation and processing delays in the TSN node. This enables the network manager to accurately model traffic and schedule transmission time slot positions and sizes. Additionally, we do not assume that the drift probability distribution follows a fixed distribution. Instead, our approach dynamically models it using various techniques.

Finally, the present problem was partially addressed in our previous work from a theoretical perspective in [13]. In that study, using traffic frame arrival captures in a station (STA) and MATLAB simulations, we proposed three mechanisms, reinforcement learning, active update, and polynomial forecasting, to update the transmission time slot position of the STA and reduce link latency and jitter.

However, the current approach advances this earlier work in several significant ways. First, in architectural terms, the DT concept has been introduced and integrated with TSN control elements. This enabled the implementation of a distributed online data-capturing system. Furthermore, the previously proposed pseudo-synchronization mechanism for time synchronization between the end node and the STA has been removed, as it was deemed unrealistic. Finally, this work includes a real implementation of a WTSN, demonstrating the feasibility of the proposed concepts.

The DT concept is increasingly seen as an enabler of network control, yet existing literature remains fragmented across the protocol stack. Most studies target the application layer but lack port-level timing control. Alternatively, research focusing on the MAC layer often omits a distributed control architecture. Regarding modeling, current works frequently analyze frame arrival distributions without considering the time component. They often assume normal distribution patterns, which fail to reflect real-world conditions. These technical gaps oversimplify the problem and limit schedule feasibility for RST.

Our work bridges these layers through three main contributions. First, we propose a Wi-Fi-based WTSN-DT management framework. This system integrates real-time measurements with modeling and scheduling to enable port-level timing control by linking virtual and physical planes. Second, we investigate three analytical and ML strategies to model cyclic traffic. By accounting for frame arrival time series, these methods provide a solution for RST. Finally, we implement a WTSN-DT element using openwifi. This setup reduces link latency by scheduling transmission slots in a real environment. These steps provide a functional data model that synchronizes control and data planes to maintain bounded latency in real-world WTSN environments.

III. WTSN-DT SYSTEM DESIGN

As shown in Figure 2, the proposed integration, highlighted in green, does not change the existing centralized configuration management defined by the TSN IEEE 802.1Qcc standard [7]. Where the Centralized User Configuration (CUC) retrieves end-node capabilities, handles application

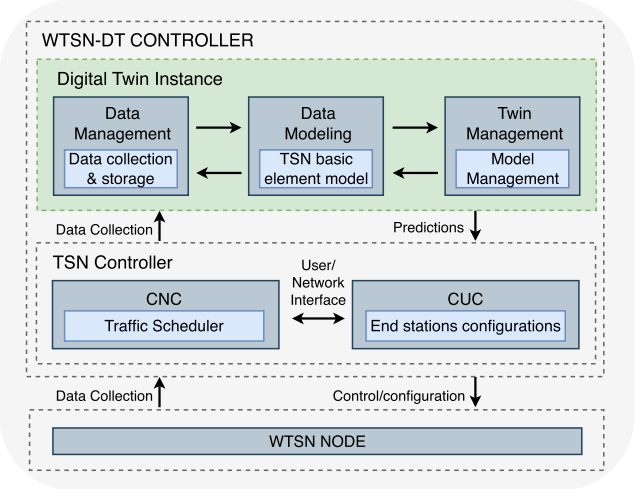


FIGURE 2. Reference architecture of a WTSN-DT.

stream requirements, and communicates stream requests to the Centralized Network Configurator (CNC), which configures end-stations according to scheduling results. Rather than altering this structure, our integration introduces the DT as an additional component. The DT acts as a virtual replica of the WTSN, encompassing devices, communication paths, and control logic. It models the behavior of devices such as STAs or APs, control flows, and network communication, enabling testing and validation.

Currently, there is no standard definition of a DT for networks. However, the IETF specifies that a digital twin for networks should include four key elements: data, mapping, models, and interfaces [20]. Real-time data is collected from physical TSN elements, and a representation of this data is generated through a data model. Two interfaces are typically defined: one between the DT and the network infrastructure, and another between the DT and applications. In WTSN, these interfaces are managed by the CNC and CUC.

In this context, the tasks of each component can be defined as follows:

- **Digital Twin (DT):** The DT functions as the system's predictive advisory layer, structured around data management, data modeling, and twin management blocks. The data management block collects and preprocesses real-time and historical WTSN data to ensure consistency and synchronization. The data modeling block maintains a high-fidelity virtual representation of the network, emulating system behavior and comparing live inputs with simulated forecasts to detect potential failures and performance bottlenecks. The twin management block coordinates model updates and scenario analyses while generating advisory outputs. Its role remains strictly analytical; it provides predictive intelligence for proactive adjustments without directly executing network changes, thereby decoupling computationally intensive modeling from time-critical operations.
- **CNC:** As the sole authority for network-layer determinism, the CNC translates DT insights into executable configurations. It manages all infrastructure-level actions, including the calculation of Gate Control Lists (GCLs), time-slot allocation, and switch configuration. By isolating scheduling logic within the CNC, the architecture ensures that network timing remains deterministic and scalable, independent of the DT's processing overhead.
- **CUC:** The CUC acts as the application-layer coordinator, aggregating requirements such as cycle times and jitter tolerance from end nodes. It relays these needs to the CNC for scheduling and to the DT for modeling. Additionally, the CUC originates synchronization commands for end devices, using DT feedback to align application tasks with network transmission windows without interfering with the CNC's infrastructure management.

This architecture marks a transition toward a complete WTSN-DT system. While the CUC and CNC retain their roles as defined in the IEEE 802.1Qcc standard, tasks such as traffic scheduling are enhanced through DT predictions. At a later stage, a full WTSN-DT system will integrate simulations to enable what-if scenarios and emulate node issues. This will allow for the prediction and resolution of failures before they occur.

IV. TRAFFIC PATTERN MODELING AND SCHEDULING

Focused on reducing the described RST problem, yet general for network modeling, the proposed and implemented WTSN DT-based control loop is shown in Figure 3. As illustrated, a traffic monitoring agent in the WTSN node captures data, specifically frame arrival times, and sends it to the WTSN-DT controller. This controller may be physically hosted on an external node, as detailed in Section V-A, in alignment with the established TSN management standard. The data is then modeled and used to generate a transmission schedule, which is sent back to the WTSN node for implementation. Therefore, this centralized topology was chosen to leverage the additional resources typically available in the controller rather than in the nodes. The next sections explain the data collection, data scaling, modeling, and scheduling blocks in detail. Before that, the RST problem is formally defined.

A. RESIDUAL SERVICE TIME (RST)

Although the RST challenge was first shown in Figure 1, its implications deserve a detailed explanation. Suppose the application frames are generated periodically but are not synchronized with the TSN TAS gating system. Also, one frame is sent per cycle, and the traffic stream has one assigned time slot each cycle. If a frame arrives just before its time slot, it is transmitted immediately. However, if it arrives after the time slot, it must wait for the next cycle.

As shown in Figure 4, $t_{arr} \in [0, T_{GCL})$ represents the frame's arrival time relative to the cycle start. The slot opens at time $t_0 \in [0, T_{GCL})$ and lasts for t_s. Then:

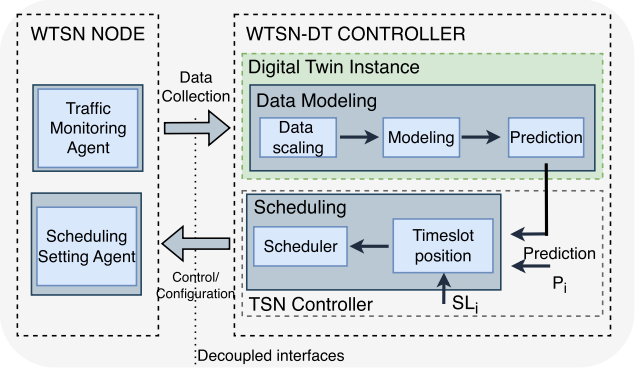


FIGURE 3. DT-based modeling and scheduling control loop.

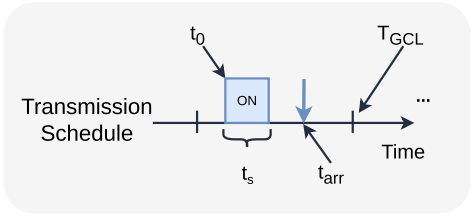


FIGURE 4. Residual service time (RST).

- If $t_{arr} \in [t_0, t_0 + t_s]$, the frame transmits immediately, and latency mainly comes from the transmission time.
- If $t_{arr} < t_0$, the frame waits for $t_0 - t_{arr}$.
- If $t_{arr} > t_0 + t_s$, the frame misses the slot and waits until the next cycle.

$$RST(t_{arr}) = \begin{cases} 0, & \text{if } t_0 \le t_{arr} \le t_0 + t_s, \\ t_0 - t_{arr}, & \text{if } 0 \le t_{arr} < t_0, \\ T_{GCL} - t_{arr} + t_0, & \text{if } t_{arr} > t_0 + t_s. \end{cases} \tag{1}$$

Consider a typical F-safe application cycle in PROFINET with a cycle time of $T_{GCL} = 10$ ms [29]. The time slot length is $t_s = 512\mu$s, and it starts at $t_0 = 2$ ms. Thus, the slot is open from 2 ms to 2.512 ms. The minimum RST is 0, while the maximum is $RST_{\max} = T_{GCL} - t_{arr} + t_0 = 9.488$ ms. This means that, depending on when the frame is generated, the application will experience an RST between 0 and 9.488 ms.

B. DATA COLLECTION

In a WTSN, STAs and APs typically serve as the entry points for traffic flows. They act as the initial interface between non-aligned frames and the TSN transmission schedule. Therefore, our approach begins by capturing frame arrivals at the WTSN traffic entrance node, specifically within the driver, just before they reach the wireless PHY. This method accounts for both the frame generation delay and the network stack processing delay of the node itself, as previously described in Figure 1. The traffic arrival timestamp of the k-th frame in a batch of M frames is represented by Equation 2:

$$\begin{aligned} t_{\text{arr}_k} &= k \cdot T_{\text{gen}} + \eta_{\text{gen}_k} + D_{\text{proc}_k} + \eta_{\text{proc}_k}, \\ &= k \cdot T_{\text{gen}} + D_{\text{proc}_k} + \eta_k, \quad \text{for } k = 1, \ldots, M, \end{aligned} \tag{2}$$

where T_{gen} represents the traffic flow generation cycle, which is either the same as or a multiple of the GCL cycle, D_{proc} is the processing time that accounts for the network stack processing delay of the k-th frame. Finally, as shown in the second line of Equation 2, the generation time jitter and the variability in processing delay are combined and absorbed into a single variable, $\eta_k = \eta_{\text{gen}_k} + \eta_{\text{proc}_k}$.

To reduce transmission overhead, the frame arrival timestamps are recorded and sent in batches. As shown in Equation 3, the M timestamp records, corresponding to frames generated every T_{gen} seconds and sampled every T_s seconds, are collected by the traffic monitoring agent.

$$M = \frac{T_s}{T_{\text{gen}}} \quad \text{subject to} \quad T_{\text{gen}} \le T_s. \tag{3}$$

Then, as shown in Equation 4, the vector $\mathbf{d}$ is created, containing the differences between the M captured frame arrival timestamps. This vector represents the frame inter-arrival times. The first difference is calculated between the last timestamp of the previous batch and the first timestamp of the current batch. Hence, the vector $\mathbf{d}$ would contain M values. For the first captured batch, the initial difference is substituted with the average values of the subsequent differences.

$$\mathbf{d} = [t_{\text{arr}_1} - t_{\text{arr}_{M_{\text{prev}}}}, t_{\text{arr}_2} - t_{\text{arr}_1}, \ldots, t_{\text{arr}_M} - t_{\text{arr}_{M-1}}]. \tag{4}$$

To reduce the data size while preserving the time component, the vector $\mathbf{d}$ and the timestamp t_{arr_1} are sent to the WTSN-DT controller. Together with the sampling rate, this data reduction strategy ensures the communication overhead remains strictly bounded.

To illustrate this process with a numerical example, assume a frame is generated every $T_{\text{gen}} = 8.192$ ms, and we sample every $T_s = 1$ s. This results in a batch of $M = 122$ captured frames per sampling interval. Assume the last frame of the previous batch arrived at exactly 100.000 ms. If the first few frames of our current batch arrive at $[108.195, 116.380, 124.580, \ldots]$ ms, the resulting inter-arrival vector is the difference between these sequential timestamps: $\mathbf{d} = [108.195 - 100.000, 116.380 - 108.195, 124.580 - 116.380, \ldots] = [8.195, 8.185, 8.200, \ldots]$.

C. DATA SCALING

Once the batch of M inter-arrival times is received at the WTSN-DT controller, it is *scaled* using equation 5:

$$\mathbf{d}_{\text{sca}} = \frac{\mathbf{d} - \mathbf{d}_{\min}}{\mathbf{d}_{\max} - \mathbf{d}_{\min}}, \tag{5}$$

where, $\mathbf{d}_{\min}$ and $\mathbf{d}_{\max}$ are the minimum and maximum values of the vector $\mathbf{d}$, respectively. Scaling can be performed either at the node or at the controller. However, because processing time is critical, scaling is handled by the controller, which typically has higher computational resources.

D. DATA MODELING AND PREDICTION

The effect of η from equation 2 on t_{arr} in the context of the GCL cycle (T_{GCL}) can be seen in Figure 5. In this example,

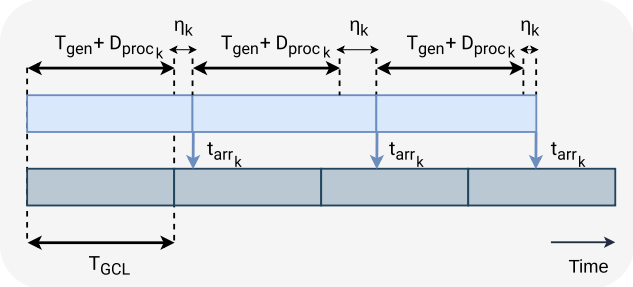


FIGURE 5. Cycle drift.

$T_{\text{gen}} + D_{\text{proc}} = T_{\text{GCL}}$, but due to the variable addition of η, a drift between the two cycles occurs. In this case, η is not treated as a cumulative measure, because each new generation depends on the one preceding it. Therefore, the goal of the methods described in this section is to model this relationship and use it to predict the subsequent frame arrivals.

For all subsequent modeling methods, the input is the scaled inter-arrival vector $\mathbf{d_{sca}}$ and the output is the corresponding scaled predicted inter-arrival vector $\hat{\mathbf{d}}$. This output is then used to compute the predicted frame arrival timestamps, as detailed in the following section. Note that each model includes internal variables that are either fixed or computed from the input.

To process these inputs and evaluate the system's predictive capabilities, we selected three distinct modeling techniques representing a spectrum of computational complexity. Linear Regression serves as a lightweight baseline to capture the primary linear trend of the short-term drift. The Kalman Filter provides an intermediate, adaptive statistical approach specifically designed to handle measurement noise and system uncertainty. Finally, a Long Short-Term Memory (LSTM) network is included as a high-complexity deep learning benchmark to evaluate whether modeling nonlinear sequential dependencies yields significant accuracy improvements. Comparing these three paradigms allows us to systematically analyze the critical trade-off between predictive accuracy and the computational overhead introduced to the network controller.

1) LINEAR REGRESSION

Starting with the simplest and most explainable approach, and assuming that most components of the captured vector $\mathbf{d}$ are linear, we propose a linear regression modeling approach. Based on the classical linear regression formulation, the matrix form of the model is expressed in Equation 6.

$$\hat{\mathbf{d}} = X\beta, \tag{6}$$

here, $\hat{\mathbf{d}}$ represents the predicted frame arrivals, while β are the learning coefficients. The design matrix X is:

$$X = \begin{bmatrix} 1 & 2 & 3 & \dots & M \end{bmatrix}^T. \tag{7}$$

Representing X in this way simplifies the forecasting process. To estimate the coefficients of β, we use the well-known ordinary least squares estimator:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{d}_{\text{sca}}. \tag{8}$$

Finally, the predicted values for the differences $\hat{\mathbf{d}}$ are:

$$\hat{d}_i = \hat{\beta}_0 + \hat{\beta}_1 \cdot i. \tag{9}$$

Then, using the cumulative sum of the *de-scaled* predicted differences, the i-th predicted arrival timestamp is:

$$\hat{t}_{\text{arr}_i} = t_{\text{arr}_1} + \sum_{j=1}^{i-1} \hat{d}_j. \tag{10}$$

2) KALMAN FILTER

The Kalman filter is a dynamic algorithm widely used in navigation and vehicle control tasks [30]. It provides an efficient computational framework for estimating the true, underlying state of a process, denoted as x_k. In our formulation, the state x_k corresponds directly to the true, noise-free scaled interarrival times. These are mathematically derived from the arrival timestamps (t_{arr_k}) defined in Equation 2. This process is modeled by the following stochastic difference equation:

$$x_k = Ax_{k-1} + w_k. \tag{11}$$

This estimation is based on measurements defined as:

$$d_k = Hx_k + v_k. \tag{12}$$

In our specific case, the measured M-scaled interarrival times (d_{sca}), where M is the scaling window size, are introduced into the filter as the measurement d_k. Because the state x_k directly represents these interarrival times, the observation matrix H is the identity matrix. By processing these measurements, the filter learns the dynamics and noise characteristics to predict the next M system states (x_k). Then, by *de-scaling* these predicted states, the expected future arrival timestamps ($\hat{t}_{\text{arr}_k}$) are calculated using Equation 10.

In Equations 11 and 12, w_k and v_k represent the process and measurement noise, respectively, with probability distributions given by:

$$\begin{aligned} w_k &\sim \mathcal{N}(0, Q), \\ v_k &\sim \mathcal{N}(0, R), \end{aligned} \tag{13}$$

where Q and R denote the covariances of the process and measurement noise. The matrix A relates the state at the previous time step $(k-1)$ to the current state (k), while H relates the state to the measurement.

The Kalman filter employs two types of equations: time update and measurement update equations. The time update (predictor) equations project the current state ($\hat{x}_{k-1|k-1}$) and error covariance ($P_{k-1|k-1}$) forward in time to generate *a priori* estimates for the next step:

$$\begin{aligned} \hat{x}_{k|k-1} &= A\hat{x}_{k-1|k-1}, \\ P_{k|k-1} &= AP_{k-1|k-1}A^T + Q. \end{aligned} \tag{14}$$

The measurement update (corrector) equations refine the *a priori* estimates by incorporating new measurements to produce *a posteriori* estimates. The Kalman gain (K_k) minimizes the *a posteriori* error covariance, as shown below:

$$\begin{aligned} K_k &= \frac{P_{k|k-1}H^T}{HP_{k|k-1}H^T + R}, \\ P_{k|k} &= (I - K_k H)P_{k|k-1}, \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(d_k - H\hat{x}_{k|k-1}). \end{aligned} \tag{15}$$

where I is the identity matrix. After each cycle of time and measurement updates, the previous *a posteriori* estimates are used to project *a priori* estimates for the next step.

3) LONG SHORT-TERM MEMORY (LSTM)

Time series data benefit from this ML approach. It learns patterns in sequential data over extended periods through gating mechanisms. Although simpler models such as GRU exist, LSTM remains the optimal starting point. Its architecture manages long-term dependencies in a virtually infinite dataset [31]. In our case, it captures the relationships between frame arrivals with greater accuracy and improves as it receives additional data batches. The model processes the input sequence to predict differences between consecutive arrival timestamps in the next batch. The LSTM model is defined as follows:

$$\hat{\mathbf{d}} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_i] = \text{LSTM}(\mathbf{d}_{\text{sca}}, \theta), \tag{16}$$

where, $\mathbf{d}_{\text{sca}}$ is the scaled input sequence, θ represents the learnable parameters, and $\hat{\mathbf{d}}$ refers to the predicted frame arrivals. Then, given the predicted frame interarrivals, the M predicted arrival timestamps are *de-scaled* and computed as shown in equation 10.

An important advantage of the three modeling cases described is that, even with inconsistent data collection, predictions can still be made for the future, as the pattern is captured by the model. While such predictions may be less accurate, they offer the benefit of decoupled interfaces, removing the need for constant frame arrival monitoring to make decisions on the traffic schedule.

E. TRAFFIC SCHEDULING

Predicting the arrival of frames allows the TSN scheduler to select the optimal position for the WTSN traffic entrance node time slot, reducing RST and, consequently, link latency and jitter. This section describes the relationship between the forecasted arrivals and the TSN schedule, which is divided into two parts: time slot position generation and schedule generation.

1) TIME SLOT POSITION

Once the N predicted frame arrival timestamps (P_i) are received in the Scheduling block, the offset of these timestamps relative to the GCL cycle can be calculated using the modulo operation $\text{mod}(\hat{\mathbf{t}}_{\text{arr}}, T_{\text{GCL}})$. The time slot position is then determined using different percentile values of the sorted modulo results.

These percentile values correspond to the required traffic flow service levels (SL). Specifically, 90%, 70%, 50%, and 40% are assigned to SL 0, 1, 2, and 3, respectively. Therefore, a lower SL provides higher confidence in capturing frame arrivals but may not yield the lowest link latency and jitter. Conversely, a higher SL offers lower confidence but results in lower latency and jitter. For this work, at the implementation, we select SL $= 0$ to prioritize capturing all predicted frame arrivals reliably, ensuring the scheduler does not miss traffic due to prediction uncertainty.

Furthermore, the relative frame arrival timestamps can be used to generate a histogram. The histogram standard deviation value, along with the transmitter Modulation Coding Scheme (MCS) value, could help define the time slot size. However, this is not considered in this work; therefore, a fixed time slot size is used.

2) SCHEDULE GENERATION

As discussed in Section I, the goal of the WTSN management framework is to determine optimal time slot positions by considering traffic flow requirements, available resources, wireless channel conditions, and factors such as frame generation. However, since reducing RST is the focus of this work, the proposed traffic scheduler optimizes only the time slot positions to minimize link latency and jitter within a single WTSN Basic Service Set (BSS).

After the system calculates the traffic flow time slot (*timeslot_start*) as described in the previous section, the scheduler generates the transmission schedule. The initial step is to identify the time slot index within the cycle. This process is detailed in Algorithm 1. The algorithm finds and returns the index closest to *timeslot_start* (*w_m*).

Algorithm 1 Timeslot Index Finder

```
procedure timeslot_index_finder(timeslot_start, gcl_cycle_idx, timeslot_size_idx)
    gen_cycle ← 512 × 2^gcl_cycle_idx
    ts_size ← 128 × 2^timeslot_size_idx
    n_timeslots ← ⌊gen_cycle/ts_size⌋
    start_positions ← {i × ts_size | i = 0, 1, ..., n_timeslots − 1}
    for i ← 0 to n_timeslots − 1 do
        if start_positions[i] ≥ timeslot_start then
            w_m ← i
            break
        end if
    end for
    return w_m
end procedure
```

Because the system frequently updates *w_m*, time slot overlaps may occur. When two time slot indices overlap, the scheduler compares their priorities to decide which one remains in place. It then shifts the other to the next available slot on the right to reduce latency and jitter. If the overlapping time slots have identical priority, the scheduler adopts a first-come, first-served rule. Finally, the scheduler sends the schedule to the Scheduling Setting Agent at the WTSN node, which then sets it.

Due to the dynamic nature of the time slots and other influencing factors, scheduling time becomes critical. However, modeling allows for the prediction of time slot positions further into the future, introducing the concept of *predictive scheduling*. This approach can be seamlessly

integrated into both current and future exact and approximate scheduling methods, such as integer linear programming, satisfiability modulo theories, or artificial intelligence-based techniques [12].

V. RESULTS

The obtained results focus on evaluating the three proposed traffic pattern modeling algorithms. In the first stage, a real-world capture of frame arrivals is used to test various algorithm configurations within a simulation environment. The frame arrival timestamps dataset is publicly available on Zenodo[2] and can be used for reproducing the results of this paper. In the second stage, the best-performing algorithm configurations are implemented and tested in a real WTSN.

Next, in subsection V-A, we describe the setup used for both data collection in the simulation stage and the real implementation in the second stage. Subsection V-B presents the simulated results, while subsection V-C focuses on the implementation results.

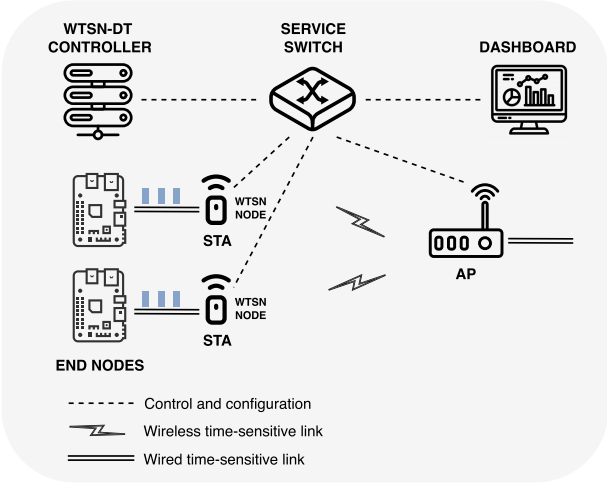


FIGURE 6. Hardware implementation topology.

A. SETUP DESCRIPTION

The proposed WTSN Digital Twin-based scheduling method has been implemented and tested using our in-house WTSN Evaluation Kit (EK), based on the openwifi IEEE 802.11/Wi-Fi baseband chip and FPGA design [21]. As shown in Figure 6, two Xilinx Zedboards equipped with AD-FMCOMMS4-EBZ transceivers act as STAs for the wireless clients and hence as WTSN Nodes, as described in Figure 3. The Access Point (AP) in our setup uses a Xilinx ZC706 board with an AD-FMCOMMS4-EBZ transceiver. Both the AP and STAs are connected via Ethernet to a service switch, which provides separate control and configuration interfaces.

Additionally, two Intel Next Unit of Computing (NUC) nodes running Ubuntu are connected to the service switch. One node manages the CNC, CUC, and DT functions, while the other serves as a monitoring dashboard. The EK also integrates key TSN features, including a Precision Time Protocol-based time synchronization mechanism and a TSN gating system within the wireless nodes [32].

The network control and configuration links are established using a ZeroMQ PUB/SUB model with agents in all WTSN nodes' userspace, serving three purposes: i) sending time-sensitive traffic arrival data to the WTSN controller, ii) setting the schedule determined by the WTSN scheduler, and iii) measuring the wireless link latency of the traffic flows. To reduce overhead, the schedule information is bitmap-encoded using the notation described in [33]. Furthermore, all nodes operated with retransmissions disabled to prevent their impact on measurements. The experiments were conducted on an empty wireless channel to eliminate external interference.

Our evaluation prioritizes the modeling approaches over operational realism. We selected a static, clean wireless channel and disabled retransmissions to ensure measurement integrity. This approach sacrifices test authenticity for enhanced measurement precision. Future work should extend the framework to evaluate the modeling approaches under retransmissions and channel loss conditions that match specific deployment scenarios.

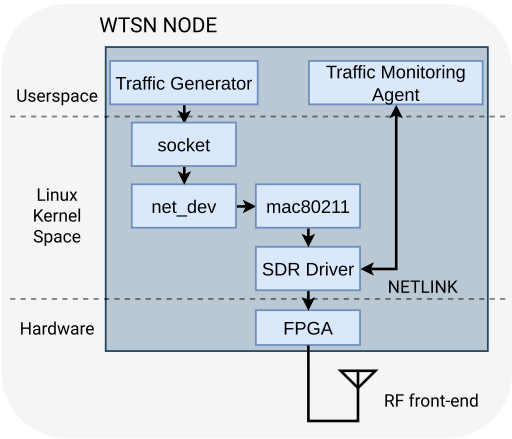


FIGURE 7. WTSN node data collection agent.

Figure 7 illustrates the schematic for traffic arrival data collection at the wireless interface of a WTSN node. The test traffic is produced by an asynchronous generator utilizing a high-resolution monotonic clock (e.g., Python's *time.perf_counter*()) to simulate non-TSN end-device behavior. Upon entering the node's kernel space, and prior to reaching the FPGA-based TSN gating system, frame arrival timestamps are captured. These timestamps are then dispatched via a NETLINK socket to the Traffic Monitoring Agent in user space. Finally, as detailed in Section IV-B, the monitoring agent aggregates these timestamps into batches and transmits them to the WTSN-DT Controller over a dedicated network control link.

If the traffic generator is located on an external node and frames enter the WTSN node via an Ethernet instance,

[2]https://zenodo.org/records/14871803

for example, these frames would also be captured and timestamped at the SDR driver. This ensures the capture of both WTSN node and generator delays.

B. TESTING RESULTS AND DATA ANALYSIS

Using the setup described in Section V-A, 100k frame arrival data points were captured from a WTSN node. These data points were then used to test the Linear Regression, Kalman Filter, and LSTM approaches in an online fashion with batches of $M = 125$ data points to identify the best methods for the next stage of implementation. For the LSTM, configurations with 2 and 3 layers of 32 neurons with 50 and 100 epochs were tested.

First, the Mean Squared Error (MSE) and Mean Absolute Error (MAE) were calculated, and the cumulative distribution function (CDF) of them is shown in Figures 8 and 9. The test compared and saved the output of N predicted elements from each model with the N actual values. In this study, we employ the CDF for most metrics, as it effectively displays achieved percentiles, a crucial factor for determinism.

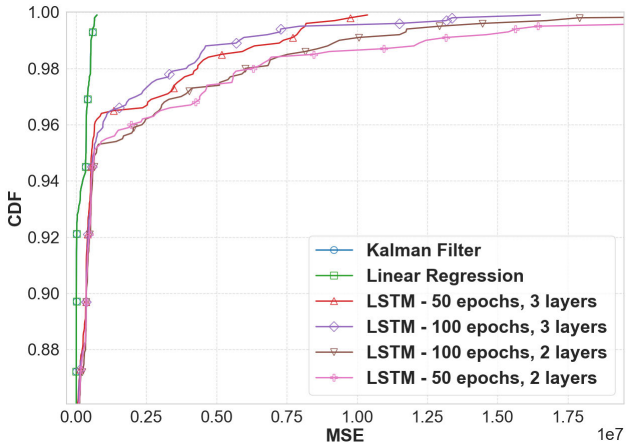


FIGURE 8. CDF of mean squared error.

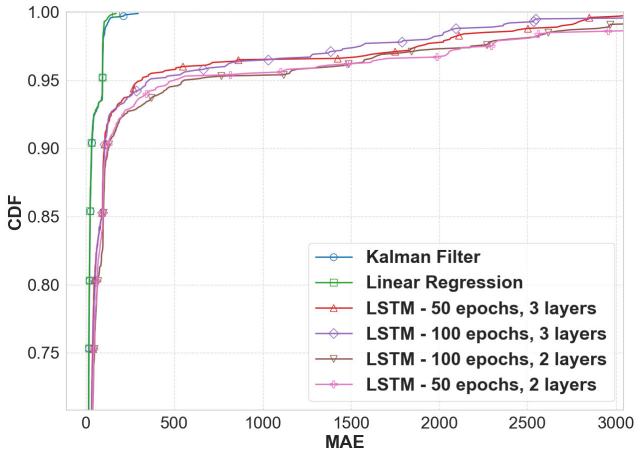


FIGURE 9. CDF of mean absolute error.

The results in Figures 8 and 9 show that both the Kalman Filter and Linear Regression outperform the LSTM across different configurations. This suggests that the drift component between arrivals is primarily linear, based on these performance indicators.

Next, to measure the fitting quality between the predicted batch values and the real ones, the Jensen-Shannon divergence (JSD) is used [34]. This method quantifies the difference between two distributions, in this case, the predicted and real distributions. It outputs a value between 0 and 1, with 0 indicating a perfect fit. As shown in Figure 10, the CDFs of the LSTM methods outperform the Linear and Kalman approaches. This was expected, as ML methods learn from the distribution shape. Specifically, for LSTM, the more epochs and layers used, the better the distribution shape is learned.

Although the LSTM performed well regarding JSD across all tested combinations, its hyperparameters require further tuning. The performance gap between point-to-point metrics, such as MSE and MAE, and the distribution-based JSD is informative. It suggests a timing lag that merits deeper investigation. This lag likely stems from the model capturing the statistical distribution of the traffic rather than the exact arrival times. While this temporal offset remains a critical factor for network determinism, this work does not treat it as a primary focus. Instead, it prioritizes the feasibility of the overall WTSN-DT framework.

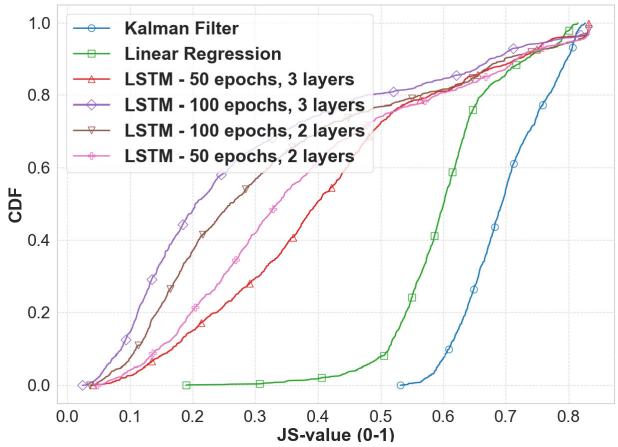


FIGURE 10. CDF of the JSD.

Effective learning often requires increased processing time and computational resources. However, in TSN, processing time is particularly critical due to its impact on time determinism. To evaluate this, we measured and compared the fitting (modeling) and forecasting (prediction) times for each iteration, as shown in Figures 11 and 12. As expected and shown in the resulting CDFs, the Linear and Kalman approaches perform similarly in terms of fitting time, while the Kalman approach is the worst for forecasting. In the case of LSTM, as anticipated, the more epochs and layers used, the higher the fitting time, while for forecasting, their performance is similar.

Crucially, these measured execution delays also serve to quantify the computational overhead introduced by the

proposed control loop. The processing time required by each algorithm directly correlates with its CPU usage, providing a clear metric of the system's computational burden.

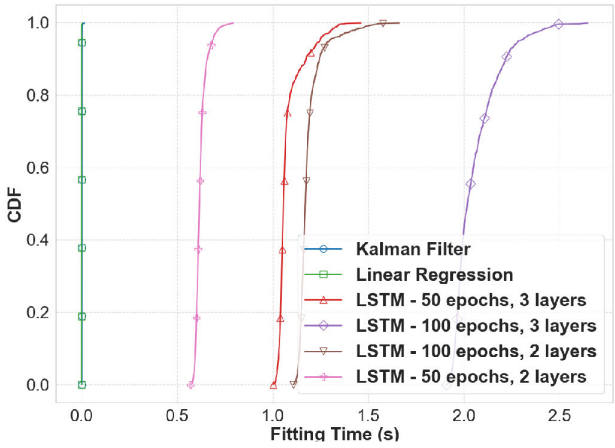


FIGURE 11. CDF of fitting time.

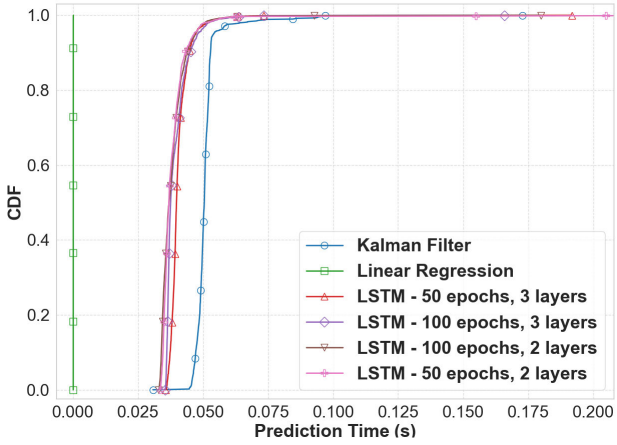


FIGURE 12. CDF of forecasting time.

Table 1 summarizes the 90th percentile performance across all metrics, providing a consolidated view of the accuracy-efficiency trade-offs. These measurements reveal a key trade-off: while LSTM could theoretically capture non-linear dynamics, its computational overhead comes at a cost. More critically, the Kalman Filter's forecasting delay (52.8 ms) significantly exceeds Linear Regression's (0.1 ms), a gap that directly impacts real-time scheduling decisions in WTSN environments. In our testbed regime, where the generation process is already well explained by linear dynamics, Linear Regression emerges as the superior choice.

C. IMPLEMENTATION PERFORMANCE EVALUATION

Based on the results from the previous section, three algorithms were implemented in the real setup: Linear Regression, Kalman Filter, and LSTM with 2 layers and 50 epochs, which provides a good balance across the evaluated indicators. Using the topology described in Figure 6, the development and testing were carried out in two stages: first, one traffic flow from a single STA to the AP was tested, followed by two simultaneous uplink traffic flows, each from a different STA to the AP. The TSN and sampling parameters used are summarized in Table 2. The experiments use $SL = 0$, which maximizes capture confidence and prevents scheduling failures under prediction variance. Figure 13 shows the real-time captured dashboard result of link latency before and after enabling the network controller, using the Linear Regression approach. As seen, once the controller is enabled, both latency and jitter are reduced significantly.

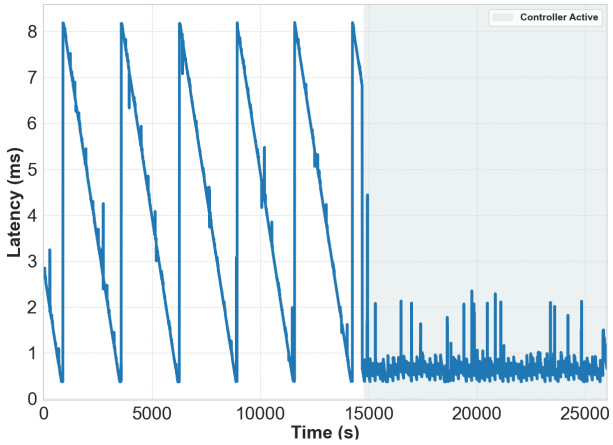


FIGURE 13. WTSN link latency control using linear regression.

An important factor to consider when implementing dynamic TSN systems is the delay introduced during data collection, fitting, prediction, schedule calculation, and setting of the schedule in the TSN node. These delays must be accounted for in the prediction process within the WTSN-DT controller before being sent to the TSN node. Additionally, as mentioned previously, the advantage of having a model is that predictions can be made for the future, but this must also be incorporated into the prediction process. Therefore, leveraging the fact that all TSN nodes are tightly synchronized, a new variable, t_{delay}, was added to the forecasting equation 10 to account for this compensation.

The t_{delay} value used in each iteration is approximated based on the previous one. However, since the WTSN-DT runs on a non-real-time operating system/kernel, t_{delay} varies. Table 3 shows the mean and standard deviation (STD) values measured in our implementation. Therefore, especially in the case of two STAs using LSTM, each traffic capture, fitting, and prediction is executed on an independent processor core to reduce this variability.

As shown in Figure 1, RST primarily occurs because frame generation is not aligned with the TSN schedule. Figure 14 demonstrates this phenomenon as measured in real life. As seen, in the *fixed schedule* case, the latency CDF, although bounded to 8,192 ms, introduces jitter. In contrast, in the *no schedule* case, the STA is allowed to transmit as soon as the frame is received (using Carrier Sense Multiple

TABLE 1. Testing results summary at 90th percentile.

Method	MSE ($x10^3$)	MAE	JSD	Fitting (s)	Forecasting (ms)
Kalman Filter	8	31.5	0.67	0.004	52.8
Linear Regression	8.5	31	0.73	0.001	0.1
LSTM - 50 epochs, 3 layers	355	103	0.72	1.17	44
LSTM - 100 epochs - 3 layers	359	106	0.67	2.22	45
LSTM - 100 epochs - 2 layers	358	126	0.7	1.24	44
LSTM - 50 epochs - 2 layers	362	119	0.72	0.65	43

TABLE 2. Measurements parameters.

Parameter	Value
$T_{\text{gen}}, T_{\text{GCL}}$	8.192 ms
Time slot size (t_s)	256 μs
Total samples	10.000
T_s	1 s
M	125
SL	0

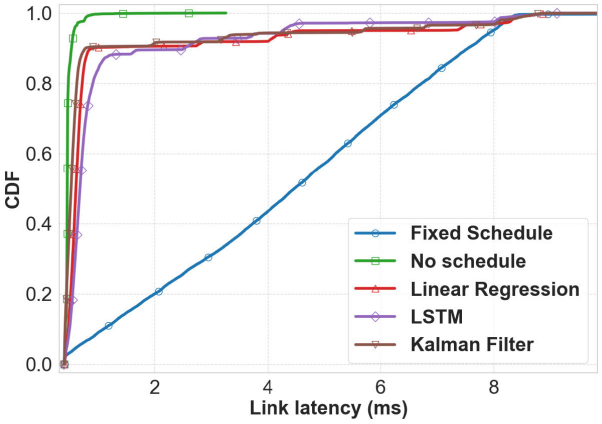


FIGURE 14. CDF of WTSN link latency for different controllers.

Access with Collision Avoidance - CSMA/CA), resulting in minimal latency and jitter. However, this approach sacrifices the TSN benefits of traffic protection and bounded latency. In contrast, the three proposed methods achieve latency and jitter levels similar to the unscheduled case while preserving the advantages of TSN. Comparing them at the 90th percentile, a 96,7% latency reduction was achieved.

An important factor, particularly for t_{delay}, is presented in Figure 15, which shows measurements of the total fitting,

TABLE 3. t_{delay} measured values.

Method	Mean Delay (ms)	STD (ms)
Kalman Filter	73.7	15.4
Linear Regression	54	2.6
LSTM	1024.3	224.5

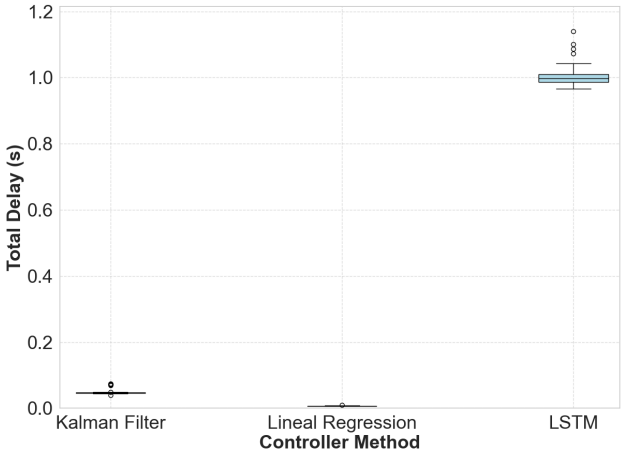


FIGURE 15. Total delay (fitting+prediction+scheduling) for different controllers.

prediction, and scheduling operations. As observed, although there are notable differences between the methods, consistent with the trends identified in the previous section, the values remain relatively constant. This consistency validates the use of the previous t_{delay} as a reliable approximation for delay compensation. Some outliers appear in the LSTM results. These arise from interference by other system processes despite dedicating a processor core. Complete isolation would require an operating system with real-time capabilities. This limitation becomes more significant for computationally intensive methods, such as LSTM.

Table 4 summarizes the 90th percentile link latency and total end-to-end delay. Linear Regression achieves the lowest total delay (47 ms), while Kalman Filter delivers the best link latency (0.77 ms), with only a marginal 14% increase in the mean delay. LSTM's link latency (2.52 ms) and especially its total delay (0.99 s) reveal the practical cost of the added complexity, directly correlating with its high t_{delay} values shown in Table 3 (mean: 1024.3 ms, STD: 224.5 ms).

TABLE 4. One STA results summary.

Method	Link Latency 90th (ms)	Mean Delay (s)
Kalman Filter	0.77	0.07
Linear Regression	0.91	0.047
LSTM	2.52	0.99

The success of these simpler models confirms that RST drift behaves as a highly linear process in the short term. While long-term clock drift in diverse WTSN environments might be inherently non-linear, driven by factors like temperature shifts and synchronization inaccuracies, our closed-loop DT architecture mitigates this through frequent sampling. Within these brief control windows, the relative drift between the frame generator and the TSN clock remains effectively linear. This continuous correction ensures the robustness of the predictive model under dynamic conditions without requiring complex, resource-heavy algorithms. However, a clear scalability trade-off emerges: maintaining this high-frequency feedback to guarantee low latency requires significant computational overhead in larger deployments.

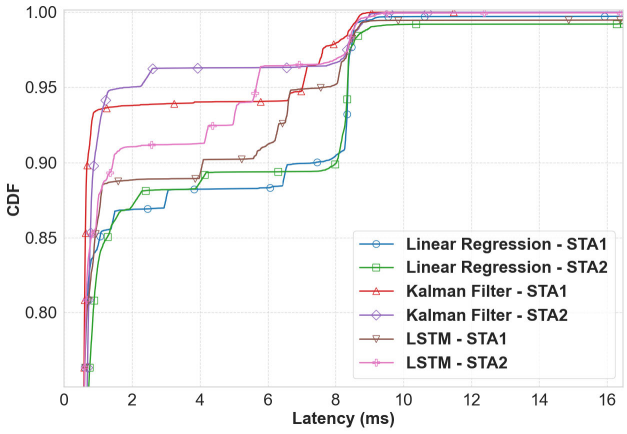


FIGURE 16. CDF of WTSN link latency for different controllers - two STAs.

Finally, to evaluate the adaptability of the system, the link latency CDF results for two simultaneous traffic flows from two STAs are shown in Figure 16. Similar to the single STA case, all approaches successfully reduce latency and jitter. However, in this scenario, the Linear Regression approach performs the worst, while the Kalman Filter delivers the best overall performance.

The performance difference between methods with one and two STAs is related to the implementation. Although the capture, fitting, and prediction processes, which introduce the highest delay, are performed on separate processor cores to achieve processing delay determinism, the scheduling is handled by a single instance. This can lead to unexpected delays. While deterministic processing is beyond the scope of this study, it plays an important role and requires further investigation.

VI. CONCLUSION AND FUTURE WORK

The wireless environment poses major challenges to achieving the determinism required by time-sensitive networking (TSN). Tasks like traffic scheduling demand enablers tailored to the unique behavior of wireless systems. Digital twins (DTs) offer a promising path forward by enabling virtual representations of network components, which support predictive scheduling. However, the interaction between DTs and TSN management remains undefined.

This work takes initial steps by proposing, implementing, and testing a wireless TSN management framework that integrates a DT focused on minimizing wireless link latency and jitter caused by residual service time (RST). Frame arrival times were modeled using linear regression, Kalman filtering, and LSTM networks to forecast arrivals and optimize transmission slot allocation. After evaluating performance using MSE, MAE, JSD, and delay metrics, the best models were deployed on our openwifi platform. While processing delay components differed across methods, all showed comparable reductions in link latency with one or two wireless clients, underscoring the effectiveness of simple models like linear regression. In this context, latency serves not only as an indicator of application-level performance but also reflects additional delays, including computational overhead.

While the proposed framework successfully reduces latency and jitter for the evaluated flows, scaling this dynamic approach to a large number of traffic flows introduces fundamental scheduling challenges. In highly dynamic deterministic wireless networks, frequent adjustments to slot positions and sizes, such as those necessitated by non-synchronized generation or Modulation and Coding Scheme (MCS) adaptations, inevitably lead to slot fragmentation and reduced resource utilization. Currently, mitigating these inefficiencies without resorting to wasteful over-provisioning remains an open research challenge across the field. Overcoming this challenge, while simultaneously maturing the predictive capabilities and environmental awareness of the Digital Twin itself, forms the foundation of our future research.

To optimize communication overhead, future work will explore variable sampling rates and extend the data-collection subsystem to include metrics such as time synchronization, channel conditions, and processing time for deeper WTSN insight. To enhance the prediction model, we will explore transfer learning and shared models for various traffic types to ensure scalability. Additionally, we plan to evaluate advanced time-series architectures, such as D-Linear, to further optimize the extraction of linear trends from highly variable network data. Furthermore, we will analyze how forecasting accuracy impacts the scheduling logic and total controller delay. By integrating contextual data like traffic and sensor information, the WTSN-DT architecture will evolve to support ''what-if'' scenarios and proactive real-time decisions. To enhance operational realism, we will introduce realistic interference patterns and variable channel conditions. We will also conduct sensitivity analysis on the Service Level parameter to explore the trade-off between capture confidence, latency, and jitter across different deployment scenarios. Finally, because the dynamics of the wireless and wired segments of a TSN are fundamentally different, we will investigate cross-domain orchestration to maintain end-to-end deterministic guarantees in large-scale, multi-hop WTSN deployments.

REFERENCES

[1] K. Anil. (2023). *Adoption Rate of Wireless TSN is Expected To Ramp Up Quickly for Industrial IoT Applications*. [Online]. Available: https://networkbuilders.intel.com/blog/adoption-rate-of-wireless-tsn-is-expected-to-ramp-up-quickly-for-industrial-iot-applications

[2] M. Ashjaei, L. L. Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems," *J. Syst. Archit.*, vol. 117. [Online]. Available: https://dl.acm.org/doi/10.1016/j.sysarc.2021.102137

[3] T. Fiori, V. Eramo, L. Yi, T. Fiori, V. Eramo, and L. Yi, "Advancing Mars rover autonomy: Integration of time sensitive networking (TSN)," *LPICo*, vol. 3007, p. 3237, Jan. 2024. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2024LPICo3007.3237F/abstract

[4] *IEEE 802.1AS-2020-IEEE Standard for Local and Metropolitan Area Networks-timing and Synchronization for Time-Sensitive Applications*, Standard IEEE 802.1AS, 2020, pp. 1–421.

[5] *IEEE 802.1QBV-2015-IEEE Standard for Local and Metropolitan Area Networks-Enhancements for Scheduled Traffic: Bridges and Bridged Networks*, Standard IEEE 802.1QBV, 2016, p. 57.

[6] *IEEE 802.1CB-2017-IEEE Standard for Local and Metropolitan Area Networks-Frame Replication and Elimination for Reliability*, Standard IEEE 802.1CB, 2017.

[7] *IEEE P802.1QCC-IEEE Draft Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, Standard IEEE P802.1QCC, 2018, pp. 1–236.

[8] K. Zanbouri, M. Noor-A-Rahim, J. John, C. J. Sreenan, H. Vincent Poor, and D. Pesch, "A comprehensive survey of wireless time-sensitive networking (TSN): Architecture, technologies, applications, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 27, no. 4, pp. 2129–2155, Aug. 2025.

[9] D. Cavalcanti, C. Cordeiro, M. Smith, and A. Regev, "WiFi TSN: Enabling deterministic wireless connectivity over 802.11," *IEEE Commun. Standards Mag.*, vol. 6, no. 4, pp. 22–29, Dec. 2022.

[10] C. Xue, T. Zhang, Y. Zhou, M. Nixon, A. Loveless, and S. Han, "Real-time scheduling for 802.1Qbv time-sensitive networking (TSN): A systematic review and experimental study," in *Proc. IEEE 30th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2024, pp. 108–121.

[11] H. Ma, H. Li, L. Chen, D. Zhao, X. Zhao, K. Xia, and G. Shou, "Time error bound of deterministic transmission in cross-domain time-sensitive networking," in *Proc. Asia Commun. Photon. Conf. (ACP) Int. Conf. Inf. Photon. Opt. Commun. (IPOC)*, Nov. 2024, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/10809464/

[12] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN)," *IEEE Access*, vol. 11, pp. 61192–61233, 2023.

[13] P. Avila-Campos, J. Haxhibeqiri, M. Girmay, I. Moerman, and J. Hoebeke, "Residual service time optimization for legacy wireless-TSN end nodes," in *Proc. 19th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Jun. 2023, pp. 466–471. [Online]. Available: http://www.ieee802.org/1/tsn

[14] T. Zhang, G. Wang, C. Xue, J. Wang, M. Nixon, and S. Han, "Time-sensitive networking (TSN) for industrial automation: Current advances and future directions," *ACM Comput. Surveys*, vol. 57, no. 2, pp. 1–38, Feb. 2025.

[15] *3GPP TS 23.501 Version 16.6.0 Release 16-System Architecture for the 5G System (5GS)*, document 3GPP TS 23.501, 2020.

[16] T. Fiori, F. G. Lavacca, F. Valente, and V. Eramo, "Proposal and investigation of a lite time sensitive networking solution for the support of real time services in space launcher networks," *IEEE Access*, vol. 12, pp. 10664–10680, 2024.

[17] M. M. Roselló and M. Lorenzo, "Smart de-jittering using network digital twin," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Jul. 2024, pp. 97–101.

[18] H. Shin, S. Oh, A. Isah, I. Aliyu, J. Park, and J. Kim, "Network traffic prediction model in a data-driven digital twin network architecture," *Electronics*, vol. 12, no. 18, p. 3957, Sep. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/18/3957

[19] (2022). *ITU-T Rec. Y.3090 (02/2022) Digital Twin Network–Requirements and Architecture*. [Online]. Available: http://handle.itu.int/11.1002/1000/11

[20] Internet Research Task Force (IRTF). (2023). *Digital Twin Network: Concepts and Reference Architecture*. [Online]. Available: https://datatracker.ietf.org/drafts/current/

[21] X. Jiao, W. Liu, M. Mehari, M. Aslam, and I. Moerman, "Openwifi: A free and open-source IEEE802.11 SDR implementation on SoC," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–2.

[22] A. Freedman. (2024). *Network Digital Twins: The Next Revolution in Network Management*. [Online]. Available: https://www.keysight.com/blogs/en/tech/educ/2024/network-digital-twin

[23] R. Tavva, "Time-sensitive networking in advanced manufacturing environments: A framework for industry 4.0 implementation," *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 6, pp. 672–678, 2025. [Online]. Available: https://al-kindipublishers.org/index.php/jcsts/article/view/10059

[24] S. Scanzio, M. Rosani, G. Formis, D. Cavalcanti, V. Frascolla, G. Marchetto, and G. Cena, "Multi-link operation and wireless digital twin to support enhanced roaming in next-gen Wi-Fi," in *Proc. IEEE 20th Int. Conf. Factory Commun. Syst. (WFCS)*, Apr. 2024, pp. 1–4.

[25] S. Robitzsch, A. de la Oliva, C. F. Chiasserini, C. E. Casetti, F. Agraz, G. Landi, L. M. Contreras, P. Szilagyi, P. G. Giardina, R. Rosales, S. Spadaro, and V. Frascolla, "Standardisation assessment of a digital twin-based multi-domain deterministic communications system," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2024, pp. 1–7. [Online]. Available: https://iris.polito.it/handle/11583/2992674

[26] S. B. Hadj Said, M.-T. Thi, M. Kellil, and A. Olivereau, "On the management of TSN networks in 6G: A network digital twin approach," in *Proc. IEEE 28th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2023, pp. 1–7.

[27] S. Sudhakaran, J. Perez-Ramirez, C. Hall, A. Regev, M. Smith, A. Stern, N. Georghiou, E. Oren, G. Kronauer, G. Venkatesan, and J. Huang, "Wireless TSN: Tests and use case demonstrations with Wi-Fi," Avnu Alliance, Tech. Rep., Jan. 2024.

[28] D. B. Mateu, J. Proenza, A. V. Papadopoulos, T. Nolte, and M. Ashjaei, "TALESS: TSN with legacy end-stations synchronization," *IEEE Open J. Ind. Electron. Soc.*, vol. 5, no. 5, pp. 807–822, 2024.

[29] J. Haxhibeqiri, P. A. Campos, I. Moerman, and J. Hoebeke, "Safety-related applications over wireless time-sensitive networks," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–8.

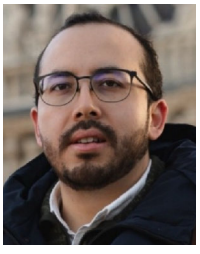
[30] G. Welch and G. Bishop, "An introduction to the Kalman filter," Dept. Comput. Science, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA, Tech. Rep., 2016. [Online]. Available: http://www.cs.unc.edu/

[31] P. Srivatsavaya. (2023). *LSTM Vs GRU. LSTM (Long Short-Term Memory) and GRU*. [Online]. Available: https://medium.com/40prudhviraju.srivatsavaya/lstm-vs-gru-c1209b8ecb5a

[32] M. Aslam, W. Liu, X. Jiao, J. Haxhibeqiri, G. Miranda, J. Hoebeke, J. Marquez-Barja, and I. Moerman, "Hardware efficient clock synchronization across Wi-Fi and Ethernet-based network using PTP," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 3808–3819, Jun. 2022.

[33] P. Avila-Campos, J. Haxhibeqiri, I. Moerman, X. Jiao, and J. Hoebeke, "Impactless association methods for Wi-Fi based time-sensitive networks," *Wireless Netw.*, vol. 30, no. 4, pp. 2457–2475, May 2024. [Online]. Available: https://link.springer.com/article/10.1007/s11276-024-03681-w

[34] P. Maurya, "Getting to know Jensen–Shannon divergence (JSD)," Tech. Rep., Aug. 2024. [Online]. Available: https://medium.com/@_prinsh_u/getting-to-know-jensen-shannon-divergence-jsd-5c96720a4434



PABLO AVILA-CAMPOS (Graduate Student Member, IEEE) received the bachelor's degree in electronics and telecommunications from the University of Cuenca, in 2017, the master's degree in wireless communications from the University of Southampton, in 2019, and the Ph.D. degree in electrical engineering from Ghent University, in 2025. He researched wireless Time-Sensitive Networking at the Internet Technology and Data Science Laboratory (IDLab), Ghent University, a joint initiative between Ghent University and IMEC. He is currently a Senior Researcher at IDLab, focusing on deterministic wireless communications in hybrid wireless-wired networks, particularly Wi-Fi. His research also explores the application of machine learning in wireless systems.



JETMIR HAXHIBEQIRI (Senior Member, IEEE) received the master's degree in engineering (information technology and computer engineering) from RWTH Aachen University, Germany, in 2013, and the Ph.D. degree in engineering computer science from Ghent University, in 2019, with his research on flexible and scalable wireless communication solutions for industrial warehouses and logistics applications. He is currently a Senior Researcher at the Internet Technology and Data Science Laboratory (IDLab), Ghent University and IMEC. His current research interests include wireless communications technologies (IEEE 802.11, IEEE 802.15.4e, LoRa) and their application, the IoT, wireless time sensitive networking, in-band networks monitoring, and wireless networks management.



XIANJUN JIAO received the bachelor's degree in electrical engineering from Nankai University, in 2001, and the Ph.D. degree in communications and information system from Peking University, in 2006. After his studies, he worked in industrial research institutes and product teams in the domain of wireless technology, such as the Radio System LabNokia Research Center (a Senior Researcher), the Devices Department, Microsoft (a Senior Researcher), and the Wireless Software Engineering Department, Apple (a RF Software Engineer). In 2016, he joined IDLab, a core research group of IMEC with research activities embedded with Ghent University and the University of Antwerp. He is working as a Postdoctoral Researcher at Ghent University on flexible real-time SDR platform. His main interests are SDR and parallel/heterogeneous computation in wireless communications. On his research track, more than 20 international patents and articles have been authored/published.



INGRID MOERMAN (Fellow, IEEE) received the degree in electrical engineering, in 1987, and the Ph.D. degree from Ghent University, in 1992. She was a part-time Professor at Ghent University, in 2000. She is a Staff Member at IDLab, a core research group of IMEC with research activities embedded with Ghent University and the University of Antwerp. She is coordinating the research activities on intelligent wireless networking (iWiNe) at Ghent University, where she is leading a team of more than 30 researchers. She is also a Program Manager of the 'Deterministic Networking' track, part of the Connectivity Program at IMEC, and in this role she coordinates research activities on end-to-end wired/wireless networking solutions driven by business-and mission critical applications that have to meet strict quality of service requirements in terms of throughput, bounded latency and reliability in private wireless environments. She has a longstanding experience in running and coordinating national and EU research funded projects.



JEROEN HOEBEKE received the master's and Ph.D. degrees in engineering computer science from Ghent University, in 2002 and 2007, respectively, with his research on adaptive ad hoc routing and virtual private Ad hoc networks. He is currently an Associate Professor at the Internet Technology and Data Science Laboratory, Ghent University and IMEC. He is author or co-author of more than 150 publications in international journals or conference proceedings. He is conducting and coordinating research on wireless (IoT) connectivity, embedded communication stacks, deterministic wireless communications, and wireless network management.