POLITECNICO DI TORINO
Repository ISTITUZIONALE

Target Wake Time Scheduling for Time-sensitive and Energy-efficient Wi-Fi Networks

Original
Target Wake Time Scheduling for Time-sensitive and Energy-efficient Wi-Fi Networks / Busacca, F., Puligheddu, C., Raviglione, F., Rusca, R., Casetti, C., Chiasserini, C.F., Palazzo, S.. - In: IEEE TRANSACTIONS ON MOBILE COMPUTING. - ISSN 1536-1233. - 25:3(2026), pp. 3469-3487. [10.1109/TMC.2025.3617324]

Availability:
This version is available at: 11583/3003470 since: 2025-09-30T06:15:38Z

Publisher:
IEEE

Published
DOI:10.1109/TMC.2025.3617324

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

05 September 2026

Target Wake Time Scheduling for Time-sensitive and Energy-efficient Wi-Fi Networks

Fabio Busacca, *Member, IEEE,* Corrado Puligheddu, *Member, IEEE,* Francesco Raviglione, *Member, IEEE,* Riccardo Rusca, *Member, IEEE,* Claudio Casetti, *Senior Member, IEEE,* Carla Fabiana Chiasserini, *Fellow, IEEE,* and Sergio Palazzo, *Senior Member, IEEE*

***Abstract*—Time Sensitive Networking (TSN) is fundamental for the reliable, low-latency networks that will enable the Industrial Internet of Things (IIoT). Wi-Fi has historically been considered unfit for TSN, as channel contention and collisions prevent deterministic transmission delays. However, this issue can be overcome by using Target Wake Time (TWT), which enables the access point to instruct Wi-Fi stations to wake up and transmit in non-overlapping TWT Service Periods (SPs), and sleep in the remaining time. In this paper, we first formulate the TWT Acceptance and Scheduling Problem (TASP), with the objective to schedule TWT SPs that maximize traffic throughput and energy efficiency while respecting Age of Information (AoI) constraints. Then, due to TASP being NP-hard, we propose the TASP Efficient Resolver (TASPER), a heuristic strategy to find near-optimal solutions efficiently. Using a TWT simulator based on ns-3, we compare TASPER to several baselines, including HSA, a state-of-the-art solution originally designed for WirelessHART networks. We demonstrate that TASPER obtains up to 24.97% lower mean transmission rejection cost and saves up to 14.86% more energy compared to the leading baseline, ShortestFirst, in a challenging, large-scale scenario. Additionally, when compared to HSA, TASPER also reduces the energy consumption by 34% and reduces the mean rejection cost by 26%. Furthermore, we validate TASPER on our IIoT testbed, which comprises 10 commercial TWT-compatible stations, observing that our solution admits more transmissions than the best baseline strategy, without violating any AoI deadline.**

***Index Terms*—Target wake time, time-sensitive networking, traffic scheduling, Industrial Internet of Things**

I. Introduction

The transformation brought about by technologies like the Industrial Internet of Things (IIoT), Artificial Intelligence (AI), and cyber-physical systems is changing the way industries and production chains operate [1]. In this new industrial paradigm, the traditional boundaries of manufacturing are redefined through smart factories, where real-time data, interconnected systems, and automation are seamlessly integrated. With IIoT at its core, this transformation ushers in a wave of intelligent sensors, actuators, and data-driven decision systems that promise unprecedented efficiency and precision in industrial processes.

F. Busacca and S. Palazzo are with University of Catania, Italy.

C. Casetti, C. F. Chiasserini, C. Puligheddu, F. Raviglione, and R. Rusca are with Politecnico di Torino, Italy.

This work was supported by the European Commission through Grant No. 101095890 (Predict-6G project), and by the EU under the Italian NRRP of NextGenerationEU, through the RESTART program (PE0000001) and the MOST CNMS (CN00000023). This manuscript reflects only the authors' views and opinions, neither the EU nor the EC can be considered responsible for them.

Reliable, low-latency communication is essential in smart factories where deployments may involve even hundreds of IIoT devices exchanging time-sensitive data to monitor production lines, detect anomalies, and coordinate actions across complex processes. As a result, communication delays and network reliability become critical, not only to maintain efficiency but also to avoid costly interruptions that could lead to production downtime, equipment damage, or even safety hazards. However, meeting these communication demands is challenging in a wireless environment, where signals are subject to interference and collisions that can delay critical messages.

While ARQ (Automatic Repeat reQuest) mechanisms can sometimes recover lost transmissions, they cannot ensure that data remains relevant. A key challenge is managing the Age of Information (AoI), i.e., the time elapsed between data generation and its reception at the destination. High AoI can make data obsolete, especially in cases where an alert about a production anomaly arrives too late for effective intervention. For example, delayed sensor data could leave operators with no choice but to halt production, leading to significant financial and operational costs.

To overcome these challenges, traditional approaches might suggest isolating each device on a dedicated wireless channel, but this is rarely feasible given the limited spectrum and high device density in IIoT settings. Thus, a more sophisticated approach is needed to manage channel access while minimizing latency and ensuring data freshness.

Target Wake Time (TWT), a feature introduced with Wi-Fi 6 (IEEE 802.11ax), offers a promising solution for managing time-sensitive communication in the IIoT. TWT allows an access point (AP) to pre-define wake-up times for each station (STA) in the network, scheduling them to transmit and receive data in dedicated slots and enabling them to sleep in the meantime. This level of control provides several critical advantages:

- Enhanced energy efficiency: Many IIoT devices run on battery power, and frequent data exchange can quickly drain their energy. With TWT, devices can remain in a low-power mode and only wake up for scheduled transmissions, significantly extending their operational life.
- Reduced channel contention: By allocating exclusive transmission times for each device, TWT helps prevent contention, reducing the chances of collisions and network delays. This allows for more deterministic commu-

nication, essential for safety-critical and real-time applications in manufacturing.
- Improved predictability and network stability: TWT scheduling makes it possible to better anticipate network load and reduce jitter, which can be vital for industrial processes that depend on synchronized device interactions and precise timing.

While TWT can address the issues of timing and energy-efficient operation in IIoT, it leaves open the question of *how to strategically assign wake times to devices* to best balance data freshness, energy savings, and throughput. Without a systematic scheduling approach, there is no guarantee that TWT allocations will meet the demanding latency and reliability requirements of time-sensitive applications. Therefore, a robust TWT scheduling mechanism becomes essential for IIoT networks that use IEEE 802.11ax. Such a mechanism should:

(i) Ensure that devices wake up only when necessary, reducing energy consumption;
(ii) Maximize the admission rate of critical and timely data flows to prevent stale or outdated information;
(iii) Minimize AoI, ensuring that time-sensitive information remains fresh and actionable.

In this work, we thus focus on making the TWT mechanism suitable to the stringent requirements of IIoT traffic by scheduling device wake times to: optimize energy saving, accommodate as many traffic flows as possible while prioritizing critical data, and guarantee a timely delivery of information. Hence, with this goal in mind, we introduce the TWT Acceptance and Scheduling Problem Efficient Resolver (TASPER), an algorithmic solution for energy-efficient, AoI-constrained TWT scheduling in Wi-Fi networks.

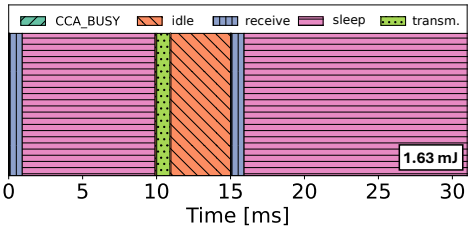
Our contributions can be summarized as follows:

- We model a TWT-enabled Wi-Fi networks, where STAs generate AoI-constrained traffic. Such a model captures the relationship between the traffic generated and transmitted by the Wi-Fi STAs, the STA power states, and the associated energy consumption.
- We build upon this model to define the TWT Acceptance and Scheduling Problem (TASP), whose objective is to minimize the rejected (therefore, not scheduled) traffic and the stations' energy consumption while satisfying the maximum AoI constraint. Finding a solution to the TASP implies determining *(i)* whether or not to admit the traffic, and *(ii)* the target wake time(s) of each STA in the network. Notably, the possibility to reject traffic may allow for a schedule with sufficient room for higher-priority traffic.
- As the TASP is NP-hard, to solve it efficiently we devise a novel heuristic strategy that efficiently produces high-quality scheduling decisions while accounting for energy consumption. The algorithm is designed to be lightweight and fast, making it suitable for timely execution in dynamic IIoT environments.
- We enhance the ns-3-based simulator for TWT by Venkateswaran et al. [2], introducing several novel features to simulate advanced TWT scheduling approaches such as TASPER. The simulator, that we call *ns-3-twt*, includes the possibility to simulate deadlines, different classes of devices and multiple TWT networks.
- We thoroughly evaluate TASPER and other baselines, leveraging ns-3-twt and show that it offers substantial improvements over the other strategies. Specifically, TASPER obtains up to 24.97% lower mean rejection cost and saves up to 14.86% more energy compared to the best baseline. Additionally, it reduces energy consumption up to 34% and the mean rejection cost of up to 26% when compared to HSA, a state-of-the-art approach originally designed for WirelessHART networks.
- We design and implement an IIoT testbed comprising 10 commercial TWT-compatible STAs. Through our testbed, we validate TWT as a means for energy-efficient traffic scheduling without incurring channel contention delays, observing 49% energy saving and 16% lower median AoI. Also, we remark that TASPER admits more transmissions than the best baseline strategy, confirming its effectiveness in scheduling transmissions.

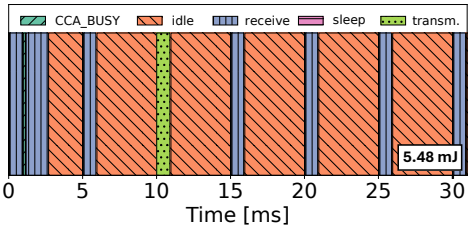
The rest of the paper is organized as follows. Sec. II highlights the advantages of the TWT feature and motivates the need for a transmission scheduling that accounts for traffic generation times and deadlines. Sec. III introduces our system model and the TWT scheduling problem, which is shown to be NP-hard. Sec. IV presents the key ideas and principles behind the TASPER scheduling algorithm we propose. After describing in Sec. V our ns-3-based simulation framework and evaluation methodology, and the scheduling baselines considered as benchmarks for TASPER, Sec. VI shows the numerical results we obtained. Sec. VII introduces the experimental testbed we set up with commercial off-the-shelf devices and validates the feasibility and good performance of our solution. Finally, Sec. VIII discusses some relevant related work, and Sec. IX draws our conclusions.

II. TWT Operations: The Need for a Scheduling Strategy

TWT enables the Wi-Fi STAs to negotiate awake periods with the Access Point (AP) to exchange traffic and to enter a doze mode during the remaining time to save energy [3]. One of the main advantages of TWT is therefore energy efficiency. Indeed, as the STAs are able to enter a sleep mode while not involved in the transmission or reception of data, they can save a substantial amount of energy that would instead be spent turning the radio into receive mode for data that is not destined to them, or in other operations such as Clear Channel Assessment [3], [4]. The advantage of using TWT in terms of energy savings, which is especially important in energy-constrained battery-powered IIoT sensors, is highlighted in Fig. 1. The plots show the output of our ns-3-twt simulator (described in Sec. V-A) in a scenario with 8 STAs, that need to send their traffic to an AP, modeling IIoT sensors of different kinds and with different energy figures. The traffic consists of a 1500-byte UDP packet encoding sensor data to be transmitted to the AP. Each STA m (m=1, . . . , 8) receives a UDP packet from its higher layers at a time equal to $(m \cdot 5)$ ms. We also consider that the transmission periodicity of the STAs is known (which is likely to happen for common



(a) With TWT



(b) Without TWT

Fig. 1. Example of the temporal evolution of STA's power states with (a) and without (b) TWT, over a period of 32 ms. The total energy consumed during the time period is reported at the bottom right. The example includes 8 STAs, each with a 1,500-byte UDP packet to transmit; the considered STA starts transmitting at time 10 ms.

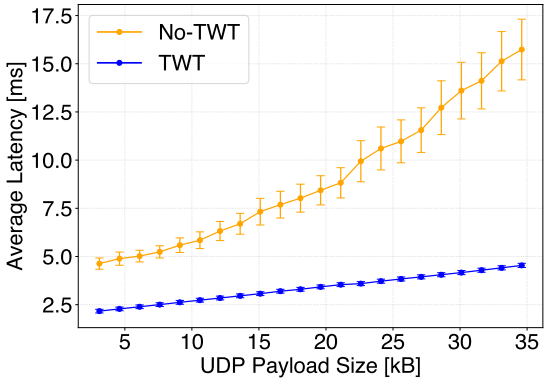


Fig. 2. Comparison of the average STA transmission delay measured with and without TWT, when used to avoid channel contention. Error bars show the 95% confidence intervals.

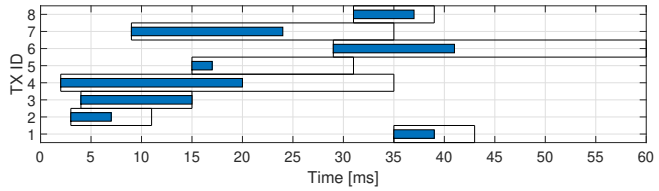
IIoT sensors that transmit their data periodically); thus, the STAs can conveniently employ a TWT mechanism in which each TWT SP starts as soon as a UDP packet is received at a STA from its higher layers. Each TWT SP is set to last slightly more than 5 ms, just for the sake of showing a clear illustrative example. Fig. 1 shows the difference with (a) and without (b) TWT, depicting the evolution of the physical layer states of STA 2 over time, and including the total energy consumption in a reference 32-ms period. One can observe that, with TWT, the STA spends most of the time in the very low energy sleep state, except for the time in which it has to receive the beacon from the AP (the receive mode right after the beginning of the simulation) and during its Service Period (SP), in which it transmits its data. Conversely, without TWT, the STA never enters the *sleep* state, and alternates between the *idle* and the *receive* state. Specifically, such states are entered every time the STA is overhearing, i.e., another STA transmits data to the AP. Since the idle state is associated with a higher energy consumption than the sleep state, employing TWT results in a significant $3.4\times$ energy saving over a reference period of just 32 ms.

Besides reducing energy consumption, TWT can enable time-sensitive Wi-Fi networks, i.e., it can provide deterministic latency for the traffic generated by the Wi-Fi nodes. Indeed, without TWT, nodes that generate traffic with different requirements and deadlines need to contend for the shared wireless medium. This is especially true when the overall offered traffic is bursty, which will make the nodes more likely to compete for channel access, and possibly defer their transmissions. This makes the timing of channel access and successful data transmissions non-deterministic. Such a level of unpredictability may be unacceptable, especially for time-critical Industry 4.0 use cases often involving the control of time-sensitive machinery. This concept is demonstrated in Fig. 2, showing the average delay experienced by 8 STAs with TWT enabled and disabled, obtained through our ns-3-twt simulator, as the payload size increases. All STAs generate traffic at the same time. By scheduling non-overlapping windows for each STA, the TWT approach yields a data delivery delay that is almost equal to the case where there would not be any contention, with lower jitter and higher determinism (the confidence intervals are indeed very small). When instead TWT is disabled, the data delay becomes much higher, and its variation very evident.

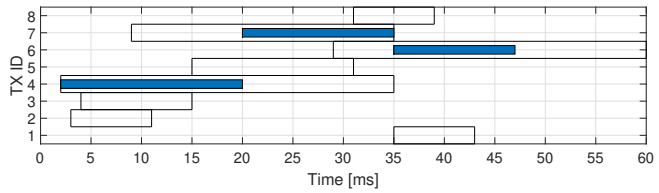
In spite of the above advantages, it is important to remark that *TWT alone does not provide any guaranteed AoI*, as Wi-Fi packets could wait in the transmission queue for more than what the maximum tolerable AoI allows, before they are scheduled in a TWT SP. This is exemplified in Fig. 3, which presents in plot (a) an instance of a problem of TWT acceptance and scheduling at the AP. For a scheduling solution to be feasible, the transmissions (blue bars) cannot overlap in time. Rather, they have to be scheduled sequentially within the white boxes, indicating the transmission ranges that respect the data deadlines. When a naive first-input-first-output (FIFO) approach is used in plot (b), only a few transmissions can be scheduled, while several are not since they would miss their deadline. In contrast, by accounting for the traffic deadlines, the number of transmissions can be maximized (see plot (c)), doubling the number of transmissions that can be accommodated. The above observations highlight that, in the presence of energy-constrained devices like IIoTs and strict latency requirements in data packet delivery, *it is imperative to develop a scheduling strategy for Wi-Fi networks that accounts for traffic generation times and deadlines*.

III. System Model and Problem Formulation

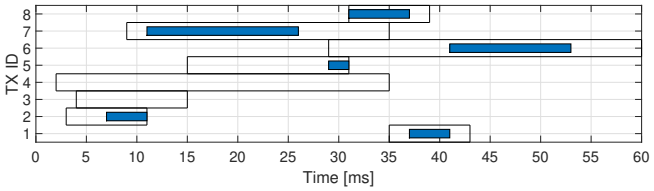
This section first introduces our system model and assumptions, and then presents the TASP optimization problem to be solved at the AP.



(a) Problem instance



(b) FIFO solution



(c) Optimum

Fig. 3. TASP example (a) and its solution using a FIFO strategy (b) and a strategy maximizing the number of transmissions (c). The blue bars indicate the transmission times, which have to start and end inside the white boxes to meet the traffic deadlines.

A. System Model and Assumptions

We consider an IEEE Wi-Fi 6 High Efficiency (HE) basic service set (BSS) with an AP and M STAs[1], with STAs having time-sensitive traffic to transmit to the AP. To avoid non-deterministic delays caused by channel contention, the AP leverages the TWT mechanism to multiplex STA transmissions in the time domain, aiming to schedule as many traffic flows as possible. The scheduling period, defined as the time between consecutive scheduling decisions, is bounded by the beacon frame, periodically transmitted every T_b seconds by the AP. Accordingly, each TWT scheduling decision holds valid till the next beacon transmission. A beacon interval is divided into D slots, each of duration $T_s{=}T_b/D$; a slot represents the temporal granularity for TWT scheduling. This implies that the awake period for a STA during which it can send/receive data, also called Service Period (SP), begins at the start of a slot and spans over a discrete number of slots. Additionally, we assume the propagation delay to be negligible.

A TWT session between the AP and a STA is composed of one SP or many periodic SPs, if the session is implicit [3]. For efficiency, we focus on implicit sessions, as they do not require new scheduling if the traffic pattern does not change. We remark, however, that such sessions can be modified at every periodic interval if the traffic pattern varies. Also, our solution applies to any of the TWT mechanism configurations foreseen by the Wi-Fi standard.

[1]We consider individual TWT sessions. However, the standard [3] also defines broadcast TWT, where the AP coordinates shared wake times for multiple STAs simultaneously. Moreover, it does not preclude the possibility of establishing TWT agreements directly between STAs, for instance in an IBSS setting.

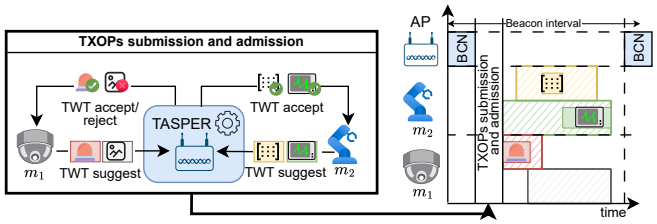


Fig. 4. System model overview. At each beacon interval, STAs first request traffic scheduling from TASPER, and then follow its scheduling decisions.

Within a beacon interval, each STA m ($m{=}1,\ldots,M$) may have one or more time-sensitive packets to transfer, possibly belonging to different traffic flows. Based on an experienced signal quality level l_m, a STA m can determine the maximum data rate $\rho_m(l_m)$ at which it can successfully transmit towards the AP. In addition to signal level, this rate depends both on the STA capabilities and on the network configuration (e.g., channel bandwidth, number of MIMO streams).

The STA then uses the TWT mechanism to request the scheduling of the data transmission for the computed transmission time, i.e., to request a *Transmission Opportunity* (TXOP) in a TWT SP. According to the Wi-Fi 6 specifications [3], the STA sends a *Suggest TWT* message to the AP requesting a desired SP, and the desired Minimum TWT wake duration needed to accommodate the TXOP. Furthermore, we assume that the STA also indicates the class of the traffic flow to which data belongs, so that the AP is aware of the corresponding traffic priority level. We will refer to the data, belonging to a given traffic flow that a STA has to transmit, simply as *transmission* (TX). The generic j-th TX is characterized by: *(i)* the source STA id m, *(ii)* the amount of data in bytes, b_j, *(iii)* the time g_j at which those data were generated at the application layer by the STA, *(iv)* the hard AoI deadline d_j within which the TX has to be received, and *(v)* the traffic priority level p_j. It follows that, for each TX j, a STA m will request a TXOP of duration equal to $\tau_j{=}b_j/\rho_m(l_m)$, where b_j denotes the amount in bytes of data belonging to a traffic flow j to be transmitted and $\rho_m(l_m)$ the achievable data rate that the STA m can use. For simplicity and without loss of generality, we express g_j, d_j, and τ_j in time slots. Note that all these elements are expressed in relative time units within the beacon interval.

The AP collects all the TXOP requests at the beginning of the beacon interval and, according to a given strategy, it schedules TWT SP for the requested TXOP by sending to each requesting STA an *Accept TWT message* (see Fig. 4). Such a message includes some important parameters, namely (i) the Target Wake Time, i.e., the time instant at which the STA should wake up for the TWT session; (ii) the TWT Wake Interval, i.e., the time interval between subsequent SPs for the STA (we assume this value to be constant for all STAs and equal to the beacon interval); and (iii) the Minimum TWT wake duration, i.e., the minimum duration that the STA should stay awake since the SP starts.

Let $\mathcal{N}$ be the set of TXOPs (or, equivalently, TXs) that the AP is requested to schedule within a beacon interval; note that

TABLE I
MAIN NOTATION

Symbol	Description
$\mathcal{N}$	Set of schedulable TXs
l_m	Quality level of the link between AP and STA m
$\rho_m(l_m)$	Maximum data rate for STA m
τ_j	No. of time slots needed for TX j
D	Number of time slots in a beacon frame
T_b	Time interval between consecutive beacon frames
T_s	Slot duration (time granularity)
g_j	Generation time (in slots) for TX j
d_j	AoI deadline (in slots) for TX j
p_j	Priority level of TX j
E_j^{tx}	Per-slot energy consumption of the STA performing TX j
E_j^{id}	Per-slot energy consumption in idle state of the STA performing TX j
E_j^{st}	Per-slot energy consumption of the STA performing TX j during mode transitions
s_{ij}	Auxiliary parameter, equal to 1 if TXs j and i are performed by the same STA; 0 otherwise
x_j	TXOP admission binary variable
y_{ij}	TXOP sequence binary variable
z_j	TXOP end time variable
e_{ij}	Total energy consumed by the STA performing TX j when j is scheduled immediately after TX i

$|\mathcal{N}|\geq M$. Then the AP will respond to each request in one of the following ways: (i) accept the TWT suggestion as is; (ii) accept the TWT suggestion with a modification; (iii) reject the TWT suggestion.

As for the per-slot energy consumption of an STA m, this is given by $E_m^{\text{tx}}=P_m^{\text{tx}}\cdot T_s$ where P_m^{tx} is the transmission power of STA m (assumed to be constant for simplicity). A TX of duration τ_j pertaining to STA m then implies an energy consumption $E_m^{\text{tx}}\cdot\tau_j$. Similarly, let P_m^{id} be the power consumed by STA m when idle, and $E_m^{\text{id}}=P_m^{\text{id}}\cdot T_s$ be the corresponding per-slot energy consumption. To save energy, before and after a TX, STA m enters the sleep state; we denote with E_m^{st} the energy consumed during the transition from sleep to transmission mode and vice versa. Let j be a TX held by STA m. For simplicity, in the following we abuse the notation by replacing E_m^{tx}, E_m^{id}, and E_m^{st}, respectively, with E_j^{tx}, E_j^{id}, and E_j^{st}. Hence, E_j^{tx}, E_j^{id} and E_j^{st} represent the per-slot energy consumption incurred, respectively, during transmission, in idle state, and while transiting between operational states, by the STA performing TX j.

B. TASP Formulation

Let us now define the TWT Acceptance and Scheduling Problem (TASP), to be solved at the AP. TASP aims to efficiently leverage the Wi-Fi TWT mechanism to meet the stringent demands of IIoT networks—specifically, minimizing energy consumption, ensuring timely data delivery, and handling concurrent traffic flows with diverse priorities and deadlines.

Ideally, the AP should schedule all the requested TXOPs, while keeping the overall energy consumption as low as possible. However, scheduling all TXs may not be possible, in which case the AP can reject one or more TWT suggestions, e.g., those associated with the lowest priority TXs, or the longest ones. Choosing *which* TWT suggestions to reject enables a trade-off between energy consumption and traffic priority level. In the following, we formulate the TWT Acceptance and Scheduling Problem (TASP), which, depending on the value of a weight coefficient, β, sets the relative importance of two objective components, namely, the cost of rejecting TXs and the STA energy consumption, and aims at minimizing both. The decision variables we consider are as follows:

- $\boldsymbol{x}=[x_j]$, defined as the *transmission acceptance vector*, where the generic integer element $x_j\in\{0,1\}$ indicates whether or not the TXOP associated to TX j is admitted;
- $\boldsymbol{Y}=[y_{ij}]$, defined as the *transmission sequence matrix*, whose element y_{ij} takes on 1 when the TXOP associated to TX j is scheduled right after the TX i, and 0 otherwise;
- $\boldsymbol{z}=[z_j]$, defined as the *end time vector* containing the end time of the scheduled TXOPs.

Additionally, we introduce two dummy TXs, denoted by α and ω, to allow for a consistent definition of the scheduling sequence of TXOPs, with TX α and ω being the first and the last of the sequence (respectively) and being both associated to null generation time, duration, and traffic priority, and an AoI deadline that is equal to the maximum among those of all genuine TXs.

We can then write the TASP as:

TWT Acceptance and Scheduling Problem (TASP)

$$\min_{\boldsymbol{x},\boldsymbol{Y},\boldsymbol{z}} \quad \sum_{j\in\mathcal{N}}\left[\beta\Big((1-x_j)p_j\Big)+(1-\beta)x_j\sum_{i\in\mathcal{N},i\neq j}e_{ij}y_{ij}\right] \tag{1a}$$

s.t.

$$\sum_{i,i\neq j} y_{ij}=x_j \qquad \forall i\in\{\mathcal{N}\cup\{\alpha\}\}\ \forall j\in\{\mathcal{N}\cup\{\omega\}\} \tag{1b}$$

$$\sum_{i,i\neq j} y_{ji}=x_j \qquad \forall i\in\{\mathcal{N}\cup\{\omega\}\}\ \forall j\in\{\mathcal{N}\cup\{\alpha\}\} \tag{1c}$$

$$z_i+\tau_j y_{ij}+d_j(y_{ij}-1)\leq z_j \qquad \forall i\in\{\mathcal{N}\cup\{\alpha\}\}\ \forall j\in\{\mathcal{N}\cup\{\omega\}\},\ i\neq j \tag{1d}$$

$$(g_j+\tau_j)x_j\leq z_j \qquad \forall j\in\{\mathcal{N}\cup\{\omega\}\} \tag{1e}$$

$$z_j\leq d_j x_j \qquad \forall j\in\{\mathcal{N}\cup\{\alpha\}\cup\{\omega\}\} \tag{1f}$$

$$e_{ij}\leq E_j^{\text{st}}+\tau_j E_j^{\text{tx}} \qquad \forall j\in\{\mathcal{N}\cup\{\alpha\}\cup\{\omega\}\} \tag{1g}$$

$$e_{ij}\geq y_{ij}\big(s_{ij}\cdot\min\{E_j^{\text{id}}(z_j-\tau_j-z_i),\ E_j^{\text{st}}\}+(1-s_{ij})E_j^{\text{st}}\big) \qquad \forall i\in\{\mathcal{N}\cup\{\alpha\}\}\ \forall j\in\{\mathcal{N}\cup\{\omega\}\},\ i\neq j \tag{1h}$$

$$z_\alpha=0,\ z_\omega=\max_{j\in\mathcal{N}}\{d_j\}\leq D \tag{1i}$$

$$x_\alpha=1,\ x_\omega=1 \tag{1j}$$

$$x_j\in\{0,1\},\ y_{ij}\in\{0,1\} \qquad \forall i\in\{\mathcal{N}\cup\{\alpha\}\}\ \forall j\in\{\mathcal{N}\cup\{\omega\}\},\ i\neq j \tag{1k}$$

In the above formulation, the cost function (1a) is the weighted sum of two elements: (i) the *rejection cost*, defined as the sum of the priority of the TXs associated with the rejected TXOPs, and (ii) the energy cost associated with the

transmission of the scheduled TXs. Specifically, the energy cost associated with TX j strictly depends on the scheduling order. Let TX i be the TX scheduled immediately before TX j, and s_{ij} be an auxiliary variable that takes on 1 if TX i and TX j are performed by the same STA, and 0 otherwise. Then we define the energy cost associated with TX j, e_{ij}, as:

$$e_{ij}=\tau_j E_j^{\text{tx}}+(1-s_{ij})E_j^{\text{st}}+s_{ij}\cdot\min\left(E_j^{\text{id}}\cdot(z_j-\tau_j-z_i),E_j^{\text{st}}\right). \tag{2}$$

Indeed, scheduling TX j obviously implies an energy consumption proportional to the duration of TX j. Additionally, the STA performing TX j incurs a cost that depends on the scheduling order: if TX i (i.e., the preceding TX), is performed by a different STA (s_{ij}=0), the STA performing TX j will need to transit from sleep to transmit state, thus incurring an energy cost E_j^{st}. Conversely, if TXs i and j are performed by the same STA (s_{ij}=1), the latter can either enter the idle state and wait to perform TX j, or go to sleep and move to transmit state just before performing TX j. Between these options, the STA will select the one that implies the minimum energy consumption. For simplicity and without loss of generality, in the model we neglect the energy consumed by a STA to receive the acknowledgment from the AP (this will instead be accounted for in our performance evaluation). Finally, note that both the priority and the energy cost of each TX are properly normalized between 0 and 1, in order to be comparable.

Constraints (1b) and (1c), instead, enforce that the scheduled TXOPs occur sequentially, without overlapping with each other. Constraint (1d) ensures that if the TXOP i precedes the TXOP j, the latter ends τ_j slots after the TXOP i. This implies that the end time of TX j must occur after the end of TX i plus the duration of TX j. Crucially, this ensures that no TX can be scheduled before one of its predecessors [5]. Constraint (1e) imposes that the end time of a scheduled TXOP does not occur earlier than the sum of the generation time and the duration of the associated TX. In other words, if a transmission is generated at time t=g_j s and requires τ_j s, the TXOP cannot end before t=g_j+τ_j s. The AoI deadline constraint (1f) specifies that the end time of a scheduled TXOP must not exceed the deadline of its associated TX. Regarding energy, (1g) upper limits the energy consumption of the STA associated with TX j to the sum of the energy required for transiting from sleep to transmit mode and vice versa, and of the energy consumed while transmitting. Conversely, (1h) provides a lower bound to energy consumption, which depends on whether the considered TXOP is requested by the same STA as the preceding TXOP. As already specified before, if the STA is the same, then such a STA remains in an idle state instead of going back to sleep, thereby conserving energy. Constraints (1i) and (1j) define $\boldsymbol{x}$ and $\boldsymbol{z}$ for the dummy TXs. In particular, the dummy TX α opens the scheduling chain while the dummy TX ω marks the end of the chain; both have a duration equal to 0 s and ω must occur before the end of the beacon interval. It follows that a feasible schedule yields a sequence of TXs that does not exceed the beacon interval. (1k) enforces the elements of $\boldsymbol{x}$ and $\boldsymbol{Y}$ to take a binary value. All notations used so far are summarized in Table I.

Importantly, based on [6], we can map the TASP into a single-machine, sequence-independent, completion-dependent batch setup cost scheduling problem with release dates, deadlines, and rejection. Indeed, the batch setup cost can represent the transceiver activation cost for one or more subsequent TXs, accounting for potential idle periods between TXs (completion-dependent), along with the subsequent deactivation cost. Notably, the most related problems are *(i)* the Order Acceptance and Scheduling (OAS) [7], which includes a job setup cost analogous to the activation and deactivation energy in TASP, and *(ii)* the single-machine Job Interval Selection Problem (JISP) [8], a special case of the OAS that focuses solely on maximizing the number of accepted tasks. Both problems can model the AoI constraints for TXs (i.e., maximum completion times for jobs); however, they leave out the energy consumption cost. It follows that, to our knowledge, a formulation equivalent to the TASP has not been previously proposed. As far as the TASP complexity is concerned, the following result holds.

Theorem 1. *The TASP is NP-hard.*

Proof. The TASP includes both integer and continuous variables, as well as one quadratic constraint (1h), and therefore it is a Mixed Integer Quadratic Constrained Programming (MIQCP) problem, which is known to be NP-hard [9]. ■

It is worth noting that, even when ignoring (1h) and the energy term in (1a), the simplified problem equivalent to the single-machine Job Interval Selection Problem (JISP) is also NP-hard [8]. While MIQCP solvers such as Gurobi and IBM CPLEX can compute the optimal solution for TASP, real-time scheduling decisions must be made on a per-beacon period basis. This requirement, combined with the limited computational resources of low-power APs, makes solving non-trivial instances of TASP impractical.

IV. The TASPER Algorithm

The NP-hard nature of the TASP problem clearly calls for the design of an efficient heuristic solution that can swiftly solve TASP even in the presence of a large number of transmission requests. We thus propose the TASP Efficient Resolver (TASPER), whose design was inspired by the BALAS algorithm, introduced in [7] to solve the OAS problem (see Sec. III). The key intuition behind TASPER is to treat the TWT-based TXOP scheduling as a best-path search within a decision graph. Also, TASPER efficiently schedules traffic data transmissions and it differentiates from previous work (including BALAS) as it effectively embeds energy saving in the scheduling problem.

Specifically, TASPER solves the TX scheduling problem at every beacon interval by implementing the following multi-step strategy. The *first step* consists in building a directed decision graph, which, as shown in Fig. 5, is composed of:

- A set of vertices, including: (i) a virtual vertex, α, representing the starting point of the sequence of TXs, (ii) a vertex for each TX to be scheduled and whose AoI deadline has not yet expired, and (iii) a virtual vertex ω, representing the end of the TXs sequence. For simplicity, given a transmission request TX j, we abuse the notation and use it also to denote the corresponding vertex;

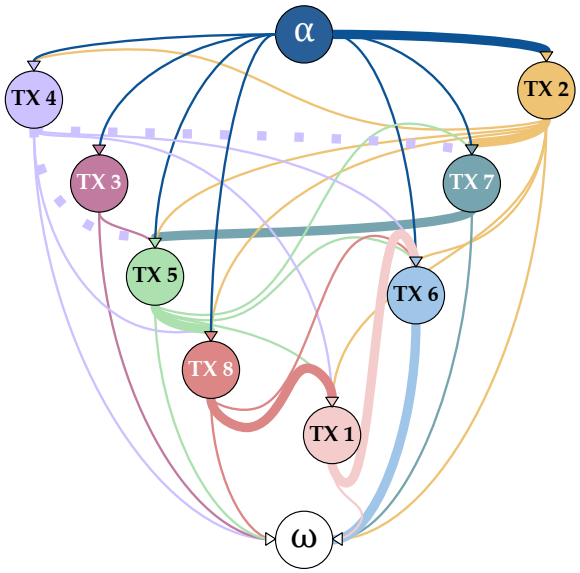


Fig. 5. Example of the TASPER graph, referring to the scheduling problem in Fig. 3. For clarity, the outbound edges have the same color as the vertex they exit while their arrow has the same color as the inbound vertex. Notice how the graph is not fully connected: e.g., there is no edge from TX 3 to TX 4, since, if TX 3 is scheduled, its end time would exceed the latest possible start time of TX 4. The solid thick lines, traversing vertices α, 2, 7, 5, 8, 1, 6, and ω, mark the best path, i.e., the one identified by the Optimum solution in Fig. 3. Finally, the dotted thick lines identify the neighborhood of Tx 4 with a neighborhood size η=1, including TXs 5 and 7.

• A set of edges, each of which connects two possible back-to-back TXs. Notice that the edges exiting vertex α and entering vertex ω are as many as the number of possible TXs to be scheduled. Also, each edge exiting vertex i and entering vertex j, is associated with a bi-dimensional weight defined as $\mathbf{w}_{ij}=[\tau_j, v_{ij}]$ where, as mentioned earlier, τ_j denotes the duration of TX j and v_{ij} is defined according to the TASP objective function, i.e., $v_{ij}=\beta p_j+(1-\beta)(1-e_{ij})$. (We recall that the priority of TX j, p_j, and the energy cost, e_{ij}, are normalized within the range $[0, 1]$).

Given the above graph, visiting a vertex means scheduling the TXOP needed to fulfill the corresponding transmission request. It follows that scheduling a sequence of TXs can be represented as a path from α to ω. Accordingly, as TASPER schedules TXOPs fulfilling the TX requests, i.e., it builds a path traversing the graph, the *second step* of the algorithm consists in identifying the feasible scheduling solutions among all the possible ones. To this end, TASPER keeps track of the residual scheduling time available within the given beacon interval. Let t_0 be the variable tracking the elapsed time. Let TX i be the last visited vertex, and TX j the potential next vertex to visit. The latter can be visited (i.e., it is a feasible choice) only if the maximum between t_0 and g_j, plus τ_j, does not exceed the AoI deadline d_j of TX j: $\max\{t_0, g_j\} + \tau_j \leq d_j$. We thus define a *feasible path* from α to ω as a sequence of edges such that, at any vertex j composing the path, the intermediate sum of the TXOP duration allocated for the upstream vertices honors the delay constraint of the TX j.

Finally, to enable TASPER to select the best scheduling solution, we define the *value* of a path as the sum of the values of the traversed edges, which is consistent with the expression of TASP's objective function. Notice that, given vertex TX j, the value taken by the weight of the edge going from TX i to TX j depends on whether the STA requesting TX i is the same that is requesting TX j. As pointed out in Sec. III, the energy consumption e_{ij} of a TX j is indeed influenced by the specific scheduling order adopted. TASPER's *third step* then consists in finding the feasible path from α to ω that maximizes the path value defined above. To reduce the time complexity of the algorithm, we let TASPER achieve this goal by considering a limited number of candidate paths. To this end, TASPER exploits the concept of *neighborhood* of a TX and uses the neighborhood size, denoted by η, as a tunable parameter to trade off the algorithm optimality gap with its time complexity. Let us consider the list of not-scheduled-yet and not-expired TXOP requests, sorted in ascending order according to their latest start time (the latest start time of a TX j is defined as $d_j-\tau_j$). Then TX j is within the neighborhood of TX i if their respective indexes in the list differ at most by η. An example of a TX's neighborhood is given in Fig. 5. In such a way, indicating with TX i the last scheduled transmission, the search for the next TX to schedule is restricted to the closest TX to TX i in the above-mentioned ordered list.

In summary, TASPER's key idea is that, at each vertex i, it selects as next TX to visit the *dominant* TX choice in the neighborhood $\mathcal{N}_i$, i.e., the one that *(i)* is associated with the highest value and that, in case of a tie, *(ii)* yields the shortest completion time. By doing so, it is possible to select the TXs that have both high priority and low completion time, thus leaving more room to schedule additional TXs. This provides a solution that maximizes the terms appearing in the TASP objective function as well as the number of scheduled TXs.

We show the pseudocode of TASPER in Algorithm 1. First, the requested TXOPs to be scheduled are sorted according to their latest start time, $d_j-\tau_j$, forming an ordered list of TXs; hence, each TX j is assigned an index ind_j corresponding to its position in this list (Lines 1–2). Then TASPER creates an empty Paths list to store the various paths from α to ω and initializes a Path for each available TX j (Line 3 and Lines 4–11). Notice that each path is initialized with a cumulative value equal to the value associated with TX j, calculated as discussed earlier in this section. Finally, the path time is set to $g_j + \tau_j$, i.e., the generation time of the TX plus the TX duration. For each path, TASPER then applies the FINDPATH procedure, as defined in Line 16. This procedure plays a key role. First (Line 17), it finds the neighbors of TX j through another auxiliary procedure, FINDNEIGHBORS, which simply applies the neighborhood mechanism previously described. As remarked above, both already scheduled TXs and any TX whose AoI deadline has expired are obviously excluded from the neighborhood. Then, for each TX n identified as a neighbor of TX j, the procedure checks whether that TX has already been visited in Path P: if so, it moves to the next neighbor TX (Lines 18–21); otherwise, the procedure includes TX n in Path P and calculates the new path P''s value and time component (Lines 22–25). Subsequently, the procedure verifies whether P' is dominated by another, already discovered path

Algorithm 1 TASPER algorithm

Input
-List of transmissions L (TX α excluded)
-Neighborhood size η
Output
-TXOP scheduling $P_{\textbf{best}}$

1: Order List L by TX latest start time
2: Assign an index ind_j to each transmission in List L, equal to the position of TX j in List L
3: Create an empty path List Pt
4: **for** TX j in L **do**
5: Create an empty Path object P
6: Create an empty list P.visited_TX
7: P.visited_TX.append(α)
8: P.visited_TX.append(j)
9: P.cumulative_reward $\leftarrow$ P.cumulative_reward + $v_{\alpha,j}$
10: P.time $\leftarrow$ $g_j + \tau_j$
11: FINDPATH(Job List L, Path P', Job j, Path List Pt)
12: **end for**
13: $P_{\textbf{best}} \rightarrow \max\{Pt, key = \text{cumulative_reward}\}$
14: **return** $P_{\textbf{best}}$
15:
16: **procedure** FINDPATH(Job List L, Path P, Job j, Path List Pt)
17: $N_j \leftarrow$ FindNeighbors(Job List L, ind_j, η)
18: **for** Tx $n \in N_j$ **do**
19: **if** $n \in P$.visited_TX **then**
20: **continue**
21: **end if**
22: $P' \leftarrow P$
23: P'.visited_TX.append(n)
24: P'.cumulative_reward $\leftarrow$ P'.cumulative_reward + v_{jn}
25: P'.time $\leftarrow \max\{g_n, P'.\text{time}\}$ + τ_n
26: **if** isDominated(P', n.paths) **then**
27: **continue**
28: **else**
29: **for** Path $P'' \in n$.paths **do**
30: **if** isDominated(P'', P') **then**
31: n.paths.remove(P'')
32: Pt.remove(P'')
33: **end if**
34: **end for**
35: n.paths.remove(P)
36: n.paths.append(P')
37: Pt.remove(P)
38: Pt.append(P')
39: FINDPATH(Job List L, Path P', Job n, Path List Pt)
40: **return**
41: **end if**
42: **end for**
43: **end procedure**

that includes the same TX n (Lines 26–28). If so, $P^{'}$ is discarded and the procedure jumps to the next neighbor TX; otherwise, any path dominated by $P^{'}$ is removed from the Path list (Lines 29–34). In such a case, the old path P is replaced by $P^{'}$ in both the lists including TX n, and the overall Path list (Lines 35–38). Finally, the procedure is called recursively until the path reaches ω (Line 39). After the procedure has identified the feasible paths, the path with the largest cumulative value is selected (Line 13).

TASPER Complexity. The time complexity of TASPER can be computed following a similar procedure to [7]. It is given by $O(n \cdot \eta^3 \cdot \sigma \cdot 4^{\eta})$, where n is the total number of TXs to be scheduled within a given beacon interval, while σ is the so-called *slack*, i.e., the maximum number of TXs ready for scheduling across the different time instants in the beacon interval. As an example, in Fig. 3, we have σ=3: indeed, after about 4 ms, and, again, at time instant 15 ms, 3 TXs are available for scheduling (namely, TX 2, TX 3, and TX 4 at 4 ms, and TX 4, TX 5, and TX 7 at 15 ms). Also, to better emphasize the scalability of TASPER with the number of STAs N, consider a given beacon interval and traffic pattern such that each STA generates P packets per beacon interval. In this case, the total number of transmissions to be scheduled is given by the number of STAs N multiplied by the number of packets P. By fixing the algorithm parameter η, the resulting complexity becomes $O(N \cdot P \cdot \sigma)$, i.e., it increases linearly with the number of STAs. For instance, as the number of STAs grows from 32 to 128, the algorithm complexity increases by a factor of four.

V. Simulation Framework and Evaluation Methodology

This section first describes ns-3-twt – the TWT ns-3-based simulation framework we developed and used for our performance evaluation. Then it introduces our evaluation methodology and simulation settings, as well as the benchmark schemes against which we compare the performance of TASPER.

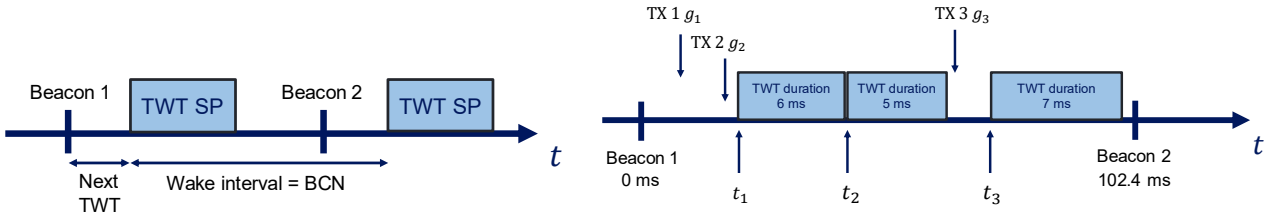
A. The ns-3-twt simulation framework

Despite the great interest received by TSN, there exist just a few open frameworks that allow for a reliable simulation of such networks, including a realistic model of IEEE 802.11ax with the TWT mechanism. Among these, to our best knowledge, the most complete existing open-source solution was the one developed by Venkateswaran et al. [2] based on the well-established ns-3 framework [10], [11], while other network simulators such as MATLAB or OMNeT++ have not been extended yet to include TWT functionalities. We therefore adopted the simulator in [2] and enhanced their solution by realizing an advanced simulation framework for TWT and TSN scenarios.

Our simulator, ns-3-twt, is released under an open-source license[2]. Building on the already existing implementation of a subset of TWT agreement messages (specifically, the implicit and individual agreements), ns-3-twt introduces a range of enhanced features, including: *(i)* a flexible, complete TWT configuration (e.g., traffic arrival time, TWT Service Period start and duration), *(ii)* the possibility to set AoI deadlines for the traffic and to log whether they are met or not, *(iii)* a customized IEEE 802.11ax simulator to easily set up simulation with IEEE 802.11ax and TWT, *(iv)* the support for multiple classes of STAs, each with a different energy model, *(v)* the seamless logging of the value and distribution of several performance metrics, *(vi)* the possibility to set up multiple TWT networks, connecting the different APs through configurable wired links.

We also highlight that a relevant feature of our simulator is the computation of the energy consumed by the STAs, leveraging an accurate energy model that differentiates the energy

[2]The link will be made available upon acceptance of the paper.



(a) ns-3-twt TWT scheduling parameters

(b) Illustrative example of a simple TWT application with 3 STAs

Fig. 6. ns-3-twt TWT implementation with the main scheduling parameters and an illustrative example of an application with three STAs that have three different transmissions (TX) with different generation times (g_j), TWT SP durations and TWT SP start times (t_j). The latter are defined through the Next TWT parameter shown on the left. Notice how the TWT scheduling is periodic unless specified otherwise, and the periodicity, also named *wake interval*, corresponds to the beacon interval (BCN).

consumed in the different PHY-layer states (i.e., transmission, reception, Clear Channel Assessment, sleep, idle). The energy model in [2] has been extended to support different classes of STAs in the same simulation. Each class has its own energy figure for each of the PHY layer states mentioned above. The simulator can report the total energy consumed in each of the PHY layer states for each STA, and several other aggregated energy metrics.

In ns-3-twt, the AP unilaterally dispatches a TWT Accept frame to all STAs based on a specified TWT schedule, to schedule their awake and sleep times. Within these frames, a crucial part is the *next TWT* field, indicating the time after the start of the next beacon interval when the STA can exit the sleep mode and access the physical channel within its TWT SP. The window lasts for a specific duration denoted by the *TWT duration*. Importantly, in our implementation, the TWT SPs are periodic. As exemplified in Fig. 6(a), this means that, unless a new schedule is provided by the AP, the STAs will periodically use the same TWT SP starting time in each beacon interval. Fig. 6(b) gives an illustrative example of a potential TWT application on ns-3-twt. The sample scenario includes an AP and three STAs, each of which has a burst of traffic to transmit; the figure highlights the traffic generation times (g_j), the TWT SP start times ($t_j{=}z_j - \tau_j$), and the respective durations for each STA within a single beacon interval.

Note that, by default, ns-3-twt considers a scenario with no other interfering networks. Thus, when scheduling only one STA at a time for each SP, no packet losses or retransmissions are expected to occur, unless the STA and the AP are too far away from each other, or TWT is not enabled. Instead, when multiple STAs are scheduled in the same SP, they may contend for the channel and their transmissions may collide. Investigating this scenario, however, is out of scope of our analysis, as we focus on individual TWT sessions. It is also important to distinguish between losses due to channel propagation conditions or channel contention, and those due to traffic that could not be scheduled by its target deadline. We remark that the latter type of losses is taken into account in the simulations.

B. Evaluation methodology

To assess the performance of TASPER, we follow three main steps: *(i)* we first focus on generating some non-trivial instances of the TASP; *(ii)* we then use TASPER, and the selected benchmarks, to derive the solutions to the instances generated in the first step; *(iii)* we evaluate the actual performance of such solutions using the ns-3 network simulator.

To generate non-trivial instances of the TASP, we started by considering three possible scenarios, with 16, 32, and 64 STAs, respectively. In all experiments, each STA j requests a TXOP associated with a single TX j. The TX parameters follow the criteria applied in [7], with parameters $\tau{=}0.2$, and $R{=}0.3$. However, since the approach in [7] accounts only for the transmission durations, in our case it is necessary to map the TXOP durations into corresponding values of Modulation and Coding Scheme (MCS) and TX frame size. To this end, the frame size is selected by sampling an empirical distribution shown in Fig. 7(a). Given the frame size, the MCS is determined as the one providing the frame transmission duration that best approximates the requested TXOP. For the generated problem instances, Fig. 7(b) and Fig. 7(c), respectively, show the resulting distributions of STA MCSs and of the TXOP durations. Then, we fixed the number of STAs and evaluated all schemes under study over 100 consecutive beacon intervals. By doing so, the TXs not scheduled during a beacon interval can be scheduled in the following beacon intervals, provided that their AoI deadline has not expired. As for running TASPER on the generated problem instances, we set $\eta{=}9$, as we found this value to offer a good trade-off between solving time and optimality gap.

Finally, as mentioned, in our simulations there are no other "interfering" networks. The real-world experimental evaluation (Section VII) instead considers a real environment including other interfering networks operating at 2.4 GHz.

C. Benchmarks

We compare TASPER against the following strategies:

- *Optimum:* it is the solution calculated by Gurobi, a well-known numerical optimization solver;
- *ShortestFirst (SF):* it prioritizes the shortest TXOP requests. At first, SF initializes the auxiliary variable $t_0{=}0$ and considers the subset of all requested TXOPs such that duration τ_j meets the condition: $t_0{+}\tau_j{\leq}d_j$ and $t_0{+}\tau_j{\leq}T_b$. This implies that the TXOP end time exceeds neither the TX AoI deadline d_j nor the end of the beacon interval T_b. Among these TXOPs, it schedules the one with the shortest requested duration. In case of ties, it selects one TXOP according to the following decision criteria (in descending order of importance): (1)

earliest TX AoI deadline, (2) highest TX traffic priority, and (3) oldest TX generation time. If any tie still exists, it selects a TXOP randomly.

• *FIFO:* it schedules the requested TXOPs according to their generation time. If, upon serving a TXOP request, the associated AoI deadline has expired, the TXOP request is dropped. Ties are solved based upon (1) the shortest TXOP duration, (2) the highest traffic priority; otherwise, the TXOP request is selected at random.

• *PriorityFirst (PF):* it works similarly to SF, but it privileges the TXOPs associated with the highest traffic priority. In case of ties, it schedules TXOP requests (in descending order of importance): (1) nearest TX AoI deadline, (2) shortest TX duration, (3) oldest TX generation time, else (4) at random;

• *Random:* it randomly schedules the requested TXOPs with duration τ_j such that $t_0+\tau_j \leq d_j; t_0+\tau_j \leq T_b$.

• *WirelessHART-based (HSA):* an adaptation of the DC-HSA algorithm from [12], originally designed for WirelessHART [13] industrial 802.15.4 networks. The original DC-HSA algorithm aims to minimize the sum of end-to-end delivery times of the scheduled traffic flows weighted by their priority level, while ensuring that each flow meets its deadline. More specifically, DC-HSA works by iteratively solving Maximum Weighted Independent Set (MWIS) scheduling problems. For each scheduled flow, DC-HSA assigns: i) a multi-hop link towards the network sink; ii) a frequency channel; iii) a transmission slot. Our implementation (HSA) preserves the core greedy MWIS scheduling logic, with three key adaptations to comply with the Wi-Fi uplink model: (i) the network topology is a single-hop star; (ii) the entire frequency channel is allocated to the STA to which access is granted; (iii) allocated time intervals are 1.024 ms long (102.4 ms beacon interval ÷ 100) instead of 10 ms as in WirelessHART.

It is also worth mentioning that, to get statistically meaningful performance results, we ran the Random strategy 100 times on each of the 100 instances. We then apply the solution to the problem instances that we obtain under the different solution schemes in our ns-3-twt simulator and derive an extensive set of results, as shown below.

VI. Numerical Results

We consider an industrial sensor network scenario, where 16 to 64 STAs (STAs) are connected to an AP on a 2.4 GHz frequency band, with SISO 20 MHz channel setup for both the STAs and the AP. The setup also employs OFDMA with a 800-ns guard interval, and customizable uplink traffic. In our scenario, the uplink traffic corresponds to packets of different lengths, depending on the problem instance. More specifically, the traffic that needs to be transmitted by each STA ranges from 1,600 B to 64,500 B, and it is fragmented into packets of 2,304 B when the total traffic that needs to be transmitted exceeds such a length (corresponding to the IEEE 802.11 Maximum Transmission Unit).

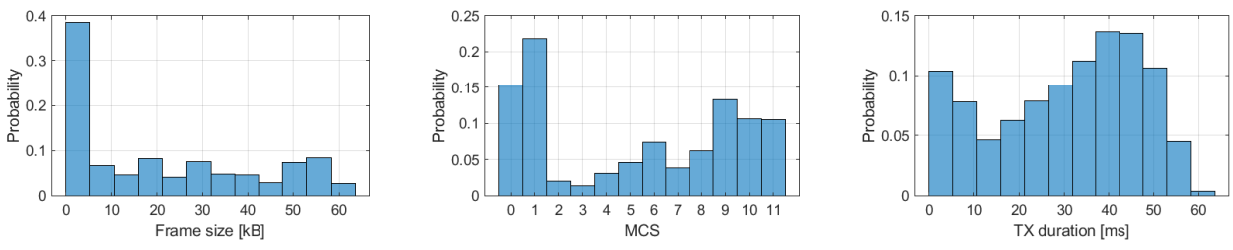
We then focus on one beacon interval of 102.4 ms, and consider that each STA may belong to one of four possible classes of STAs, each class characterized by different energy parameters, as presented in Table II.

As mentioned, we tested TASPER and the benchmark strategies on 100 different problem instances (with Random being run 100 times on each instance to guarantee the statistical validity of the experiments). The final results have then been obtained by averaging the output of each problem instance (in terms of mean rejection cost, total energy, and other metrics), and by computing the 95% confidence intervals, shown in the figures as black vertical bars.

We now consider as performance metrics the rejection cost and the mean energy consumption, where we recall that the former is defined as the cumulative sum of the traffic priority of rejected (therefore, not scheduled) TXs while the latter is computed over all STAs in the network. We remark that the unscheduled STAs will remain in sleep for the entire considered period[3]. Figures 8 and 9 present the behavior of such metrics under the different strategies, as the number of STAs varies. It is worth remarking that, due to the very high complexity level of the solution, the results for the Optimum strategy could be computed only for 16 STAs, and, even in this case, the time needed to obtain a solution through a popular solver like Gurobi greatly exceeded the beacon interval duration. This further highlights that computing the Optimum is unfeasible in a practical scenario, fostering the need to devise fast and efficient heuristics like TASPER.

Looking at the mean rejection cost in Fig. 8(a), one can observe that TASPER provides solutions that are very close to the optimum, with a mean rejection cost just 0.04% higher than Opt for β=0.9. Changing the value of β for TASPER, it is possible to obtain a mean rejection cost that varies from 3.6 for β=0.1, to 3.2 for β=0.5, reaching 2.9 for β=0.9. The latter

[3]When a STA does not receive a SP assignment, it remains in sleep mode until the next scheduling period (in our formulation, the beacon interval). To improve efficiency, STAs do not need to resubmit their scheduling requests at the beginning of each scheduling period; instead, the AP considers the requests as valid until their deadline expires.



(a) (b) (c)

Fig. 7. Distribution of: the frame size (a), the best value of MCS that can be used for TXs (b), and the TXOP duration (c), for the generated TASP instances.

TABLE II
TYPICAL PEAK CURRENT DRAW IN MA IN DIFFERENT OPERATIONAL MODES, ASSOCIATED WITH THE CONSIDERED STA CLASSES

STA class	idle	CCA_BUSY	receive	transm.	sleep
1	50	50	66	232	0.12
2	40	40	40	140	0.004
3	358	358	472	573	12
4	294	294	388.4	555.29	11.63

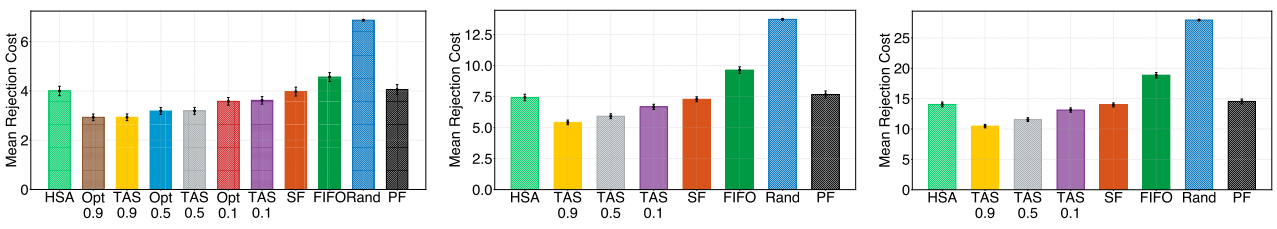
corresponds to a reduction of 19.4% with respect to β=0.1. Although ShortestFirst and HSA also give good performance, they still fall behind TASPER by 9% and 10%, respectively, even when considering the worst performance yielded by TASPER (i.e., for β=0.1). Similarly, TASPER outperforms HSA by achieving up to 10% reduction in the mean rejection cost. In fact, unlike HSA, TASPER can strategically delay high-priority flows with looser deadlines to accommodate lower-priority ones with tighter deadlines, provided all deadlines are still met. This feature allows TASPER to minimize rejected transmission requests more effectively than HSA, which schedules transmissions greedily without exploring such trade-offs.

Among all benchmark strategies, Random is the one that performs worst, as it does not employ any logic to maximize the number and priority of accepted TXOPs, nor does it take into account energy consumption. Comparing the case with 16 STAs to the more complex scenarios with 32 and 64 STAs (Fig. 8(b) and Fig. 8(c), resp.), one can notice how the overall mean rejection cost increases, since the higher the number of TXOPs to be scheduled, the higher the probability that some of them cannot be accommodated. However, the trend exhibited by the mean rejection cost remains consistent across all scenarios, and TASPER outperforms its benchmarks with a gain of 25% over ShortestFirst, 26% over HSA, 28% over PriorityFirst, 44% over FIFO and 63% over Random, for β=0.9 and 64 STAs. When considering β=0.5 to improve energy consumption, and keeping 64 STAs, TASPER still outperforms its baselines, reducing the mean rejection cost by 18% with respect to ShortestFirst and 59% with respect to Random.

Concerning the mean energy consumption, shown in Fig. 9, this metric has been accurately calculated through ns-3-twt, as mentioned in Sec. V-A. Moreover, we recall that we can obtain both the total energy consumed by the STAs in each PHY layer state (as depicted in Fig. 1), as well as the total energy consumption of each station, during the whole simulation time (shown in Fig. 9). The first important result is evident for all considered numbers of STAs. As expected, the value of β substantially affects the energy consumption: higher values of β lead to lower mean rejection cost, but at the price of a slightly increased energy consumption. Nevertheless, it is worth noting that, for both β=0.5 and β=0.1, the total energy consumed by TASPER is still lower than that of the best performing benchmark, namely, ShortestFirst. The only exception is for β=0.9, when TASPER almost completely disregards energy consumption. Looking at the case with 16 STAs, moving from β=0.1 to β=0.9 leads to a 20% energy consumption increase (from 14.3 mJ to 17.4 mJ), while in the case of 64 STAs it grows by 75% (from 6.4 mJ to 11.2 mJ). Notice, however, how the higher the number of STAs, the lower the mean per-STA energy consumption, as each STA will remain in sleep mode for a longer time. Under TASPER, this adds to the fact that, aiming at maximizing the number of scheduled transmissions, TASPER tends to select shorter transmissions as the number of STAs increases. It follows that, even when active, the time spent by a STA in TX mode becomes shorter.

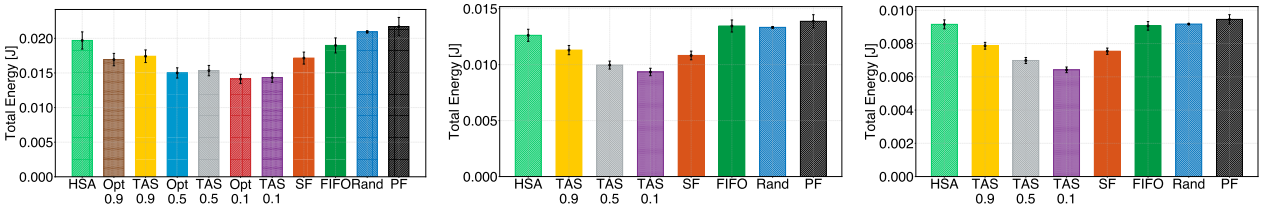
It is worth noting that, although TASPER primarily aims at maximizing the cumulative value of scheduled transmissions, it does so without violating traffic deadlines. This can be observed by looking at Table III, presenting the percentage of deadlines that are missed by the different TWT scheduling algorithms, in a scenario with 16 STAs and a value β=0.9 for TASPER. Notably, TASPER, Shortest First, and HSA consistently complete transmissions within their deadlines, thus resulting in a zero packet loss. In contrast, other strategies such as FIFO exhibit a non-negligible percentage of deadline violations. Also, it is worth remarking that, although Shortest First, HSA and TASPER all ensure that deadlines are met, TASPER outperforms its alternatives in terms of mean rejection cost and energy savings, as shown above.

Moreover, the packet loss rate is zero in all simulations, as the channel is always fully available to the single STA awake in each time slot. Indeed, on the one hand, TASPER computes the transmission duration starting from the packet size and the channel capacity. On the other hand, in the considered scenario the total generated traffic can be scheduled successfully when compared to the channel capacity. This lets TASPER schedule all the transmissions, leading to a null packet loss in the analyzed operational conditions. The



(a) 16 STAs (b) 32 STAs (c) 64 STAs

Fig. 8. Mean rejection costs, with 95% percentiles, for TASPER and its benchmarks, considering progressively complex scenarios, from (a) 16 STAs, to (b) 32 and (c) 64 STAs. From left to right, we compare TASPER (TAS) to HSA, Optimum (Opt), ShortestFirst (SF), FIFO, Random (Rand), and PriorityFirst (PF) strategies. The numbers under "Opt" and "TAS" represent the value to which we set β.



(a) 16 STAs (b) 32 STAs (c) 64 STAs

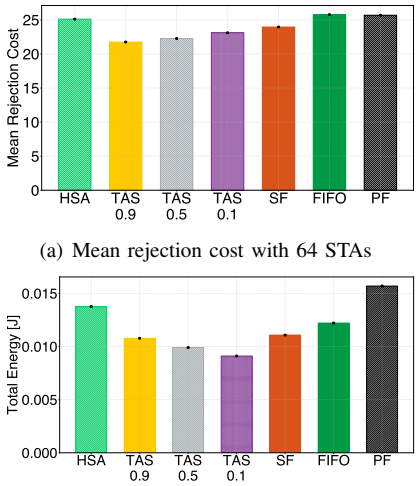
Fig. 9. Mean total energy consumption, averaged over all the STAs that transmitted at least one packet, with 95% confidence intervals, for TASPER and the different benchmarks, considering progressively complex scenarios, from (a) 16 STAs, to (b) 32 and (c) 64 STAs. From left to right, we compare TASPER (TAS) with HSA, Optimum (Opt), ShortestFirst (SF), First In First Out (FIFO), Random (Rand) and PriorityFirst (PF) strategies. The numbers under "Opt" and "TAS" represent the value of β.

TABLE III
DEADLINE MISS PERCENTAGE FOR TASPER ($\beta = 0.9$) AND ITS BENCHMARKS

	Deadline miss percentage (%)
FIFO	0.52
Priority First (PF)	0.16
Random (Rand)	0.04
Shortest First (SF)	0.00
HSA	0.00
TASPER $\beta = 0.9$ (TAS 0.9)	0.00

latter have been selected to properly assess the capability of TASPER to meet the deadlines when compared to other baselines. As a matter of fact, even the optimal solution would lose packets under different operational conditions that would not allow any feasible scheduling compared to the channel capacity.

Next, we focus on the trade-off between mean energy consumption and mean rejection cost. We can observe that a good trade-off can be achieved for β=0.5, although, comparing TASPER with β=0.1 (i.e., the case in which TASPER yields the worst performance) to ShortestFirst, TASPER still exhibits a 16% gain with 16 STAs, 14% with 32 STAs, and 15% with 64 STAs. Also, it is important to observe that the reason why ShortestFirst consumes less energy than the other benchmarks is because it selects shorter transmission opportunities, which, thanks to shorter TWT SPs, cumulatively require less energy. As for the rejection cost, Random remains the worst performing baseline, together with PriorityFirst. For instance, Random consumes 31% more energy than TASPER with β=0.5 and 64 STAs. In this case, PriorityFirst does not perform well, as it always schedules the TXOPs with the highest priority (thus providing a good mean rejection cost), but totally disregarding the fact that long TXOPs lead to high energy consumption. When compared to HSA, TASPER yields a 22% lower total energy consumption for β=0.5 and 16 STAs, and 10% reduction in mean rejection cost in the same scenario. The main reason for this performance gain lies in the energy-aware design of TASPER, which explicitly addresses the priority–energy trade-off during scheduling, favoring energy-efficient transmissions whenever none of the traffic constraints are violated. In contrast, HSA schedules transmissions based on traffic priority and time deadline alone, without accounting for energy consumption. Finally, comparing TASPER to the Optimum again in the case of 16



(a) Mean rejection cost with 64 STAs

(b) Mean total energy consumption with 64 STAs

Fig. 10. Mean rejection cost (a) and mean total energy consumption (b), along with 95% confidence intervals, in the concatenated scenario with 64 STAs. From left to right, TASPER (TAS) is compared against HSA, ShortestFirst (SF), FIFO, and PriorityFirst (PF) strategies. The numbers under "TAS" represent the value of β.

STAs (Figures 8(a) and 9(a)), it is evident that TASPER yields an energy consumption just slightly higher than Optimum, while providing an efficient scheduling solution in a much shorter time. As an example, for β=0.5, TASPER falls behind Optimum by just 1.4%.

Fig. 10 depicts the performance of the different strategies over a more complicated scheduling scenario, including 100 consecutive inter-beacon intervals. In this case, transmissions that are not scheduled in one inter-beacon interval are not rejected, rather they are reconsidered in the next inter-beacon interval, unless their AoI deadline has expired. Also, now TASPER is compared only against ShortestFirst, FIFO, and PriorityFirst. Indeed, computing the Optimum in a larger case than that with one inter-beacon interval and 16 STAs is impractical, while Random is omitted due to the poor performance it exhibited in the previous scenario.

The results in Fig. 10 further highlight the benefits of the TASPER approach, especially when considering energy efficiency and mean rejection cost jointly. Looking at Fig. 10(a), we notice that the mean rejection cost is significantly higher when compared to the performance in the previous, simpler

scenario. When β=0.9, TASPER shows substantial advantages over its benchmarks, achieving a 16% (9%) reduced rejection cost compared to FIFO (ShortestFirst). Even when considering a smaller value of β, TASPER still outperforms its alternatives, with a 10% (4%) reduction in rejection cost compared to FIFO (ShortestFirst), for β=0.1. TASPER also outperforms HSA with a 13% reduction in rejection cost, showing how our algorithm can outperform state-of-the-art solutions optimized for priority-based scheduling.

When focusing on energy consumption, Fig. 10(b), TASPER with β=0.1 leads to the highest energy savings, achieving a 18% reduction in energy consumption compared to ShortestFirst, 34% reduction relative to HSA and, remarkably, a 42% reduction against PriorityFirst. This confirms the ability of TASPER not only to serve a higher number of transmissions, but also to significantly improve the overall energy consumption. The case for β=0.9 is even more insightful. The experiments over a single inter-beacon interval showed that, when β=0.9, TASPER consumes more energy than ShortestFirst, as its main objetive is to minimize the mean rejection cost. Thus, one may wonder what leads to a different trend in such a more complex scenario. The answer lies in the fact that TASPER is able to handle the postponed transmissions very effectively, much better than ShortestFirst, ultimately achieving a lower energy consumption even if energy saving is not its primary goal.

Finally, Fig. 10 again underlines the importance of selecting a value of β that represents a good balance between rejection cost and energy consumption. Actually, TASPER with β=0.1 yields an energy consumption that is 15% lower than TASPER with β=0.9 and 8% lower than TASPER with β=0.5. However, it also exhibits a higher rejection cost (by 6% w.r.t β=0.9 and 4% w.r.t. β=0.5).

VII. Experimental Validation

To validate the observations and the proposed approach in real-world operational conditions, we now show the benefits of our approach in delivering time-critical traffic and in saving energy, using our Wi-Fi IIoT testbed built using commercial TWT-capable STAs and AP. In the following, we start by introducing the experimental testbed we designed (Sec. VII-A). Then, we present the configuration employed to first compare TWT against NoTWT, and the outcomes of such a comparison (Sec. VII-B). Finally, we analyse the performance measured with TASPER versus ShortestFirst, which, from the performance evaluation in Sec. VI, resulted to be the most performing alternative among the considered benchmarks (Sec. VII-C).

A. Our Wi-Fi IIoT testbed

To represent a real-world IIoT deployment, we selected commercial components and built a testbed comprising 10 STAs and 1 AP, as depicted in Fig. 11(a). Specifically, the STAs are Espressif ESP32-C6-DevKitC-1, accessible IoT boards that can be purchased for less than 10 USD. They are based on the ESP32-C6 System on Chip, featuring a 32-bit RISC-V processor, Wi-Fi 6 connectivity (limited to 20-MHz channel in the 2.4 GHz band, SISO), and support for Individual TWT. These boards can be programmed using the open-source Espressif IoT Development Framework (ESP-IDF), which allows for firmware coding and flashing. ESP-IDF documentation includes an application example for TWT, which has been used to develop the firmware for configuring and running the experiments. The AP is a Synology WRX560, a TWT-compatible Wi-Fi 6 router, which has been configured to assign static IP addresses to the STAs. This simplifies the board configuration since a single firmware can be flashed in all STAs; then, the desired behavior of each STA (e.g., the transmission schedule) is inferred based on the assigned IP address. Finally, to collect experimental data, an Ubuntu 22.04 host, acting as the traffic-receiving endpoint, is connected to the router via the 1 GigE LAN interface. It integrates an Intel AX200 802.11ax transceiver, which we use to monitor and timestamp frames exchanged in the Wi-Fi network. To sniff traffic, we employ the *airmon-ng* and *Wireshark* tools. As the network is secured with WPA2, to access the contents of the transmitted frames, which is useful for characterizing the data traffic, the latter is configured with the Pre-Shared Key to decrypt the AES-encrypted traffic sent by STAs. Thus, data are obtained by merging the information derived from the received application traffic and the sniffed frames.

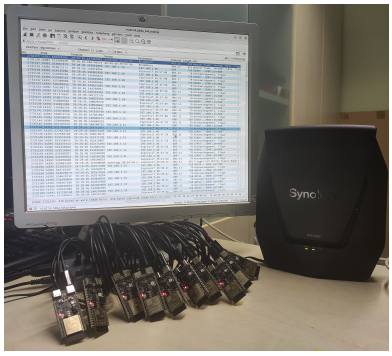
B. TWT vs. NoTWT: Experimental setup and results

To demonstrate the benefits of employing TWT to schedule traffic transmissions, we coded an application to make the boards transmit one TX of 4,800 bytes each (split in 8 frames of 600 bytes payload) to the Ubuntu host once every 10 beacon intervals (102.4 ms). To increase network saturation, the allowed MCS is restricted to MCS0 (PHY rate of 8.6 Mb/s). To avoid possible overlaps with the AP beacon, which could increase transmission delays, all boards generate data 8.192 ms after the Target Beacon Transmission Time (TBTT). This delay is motivated by the AP's behavior: we observed that at each beacon interval, the AP issues two consecutive beacons (the second one with a hidden SSID), each requiring up to 3.5 ms of airtime. All boards synchronize their clocks with the AP using the Wi-Fi timing synchronization function (TSF), which the Wi-Fi driver exposes to the application generating traffic through ESP-IDF. To avoid ARP traffic, the Ubuntu host MAC and IP addresses are stored in the boards' firmware. We tested two configurations:

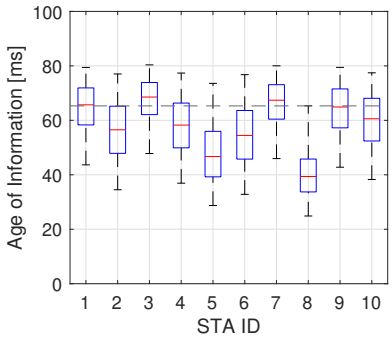
- **NoTWT**, where every board immediately tries to access the channel and transmit traffic as soon as data is generated, as in Wi-Fi networks where TWT is not employed;
- **TWT**, in which each STA has set up a TWT agreement with the AP for a non-overlapping TWT SP. Specifically, in ascending ID order, every STA SP starts 9.42 ms after the previous one. Allowing such a time interval avoids any overlap between subsequent TXOPs, with each TXOP including also the Block ACK and possible retransmissions, while making all STAs transmit within a beacon interval.

Each configuration has been tested for more than 12 hours. During this time, we captured more than 35M frames, corresponding to more than 48K transmissions by each STA.

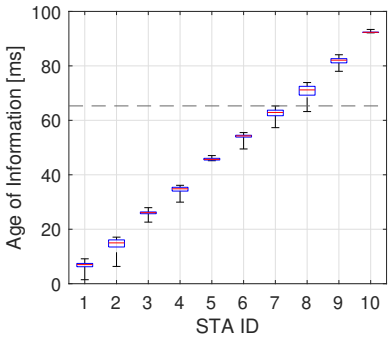
AoI measurements. The results we obtained are presented in the box plots in Figures 11(b) and (c), which show the AoI



(a) Experimental testbed. A 10-port USB hub powers the ESP32 boards. On the screen, a live Wireshark capture shows the sniffed frames. On the right, the Synology AP agrees on the TWT SPs and routes the Wi-Fi traffic.



(b) NoTWT scenario. The measured AoI is reported for each STA. STAs coordinate through the DCF (i.e., using random backoff), which leads to high AoI.



(c) TWT scenario. The measured AoI is reported for each STA. STAs have dedicated TWT SPs during which they have exclusive availability of the wireless medium.

Fig. 11. Experimental testbed and TWT vs. NoTWT results. On each box of the box plots, the central mark indicates the median, while the bottom and top edges of the box indicate the 25th and 75th percentiles (resp.). The whiskers extend to the 5th and the 95th percentiles. The horizontal dashed lines identify the AoI threshold not attainable by 95% of transmission by all STAs in the NoTWT scenario

measured for each STA in the NoTWT and TWT scenarios, respectively. We recall that the AoI measures the interval between the time of data generation at the sender and the time of its reception at the receiver.

When TWT is not used, the AoI varies significantly because of the random backoff: on average, the median AoI is 58.2 ms and its standard deviation is 12.6 ms. More importantly, we observe that none of the STAs is able, for more than 95% of the times, to send its traffic within 65.3 ms from the instant when it was generated (horizontal dashed line in Fig 11 (b)). Note that a complete transmission comprises 8 frames, which are not transmitted sequentially. In fact, between a frame and the next one, other STAs gain access to the channel, which delays transmission completion for all STAs. Thus, these results clearly demonstrate that the NoTWT configuration is unsuitable for time-sensitive networking.

Conversely, under the TWT mechanism, STAs show consistent AoI performance, according to their time schedule, thanks to the allocation of TXOPs by the AP. On average, the median AoI is 49.1 ms and its standard deviation is 2.5 ms, both lower than those of the NoTWT scenario (16% and 80% lower, respectively). Here, STAs send all of their frames before another STA attempts to do the same. Because of that, STAs with stricter AoI deadlines can be scheduled first, while those with looser AoI deadlines can be left for later. As an example, STA 1 completes its transmission within 9.2 ms more than 95% of the time. Remarkably, this approach works well for a significant number of STAs, as 7 out of the 10 STAs are able to transmit with an AoI consistently (i.e., for more than 95% of the times) lower than 65.3 ms. We recall that in NoTWT scenario no STA could achieve the same 65.3 ms AoI for 95% of the times.

Average power consumption measurements. Not only does TWT improve the AoI performance, but it also allows for energy saving as STAs can power off the Wi-Fi transceiver when they are dozing. To estimate the energy gain brought by TWT, we here refer to the average power consumed in the NoTWT and TWT cases.

First, we estimate the average power consumption in the two cases using the datasheet of ESP32-C6-WROOM-1 [14], i.e., the module comprising the ESP32-C6 chip, clock oscillator, flash memory, and PCB antenna that are installed in the ESP32-C6-DevKitC-1. It reports a current consumption of 251 mA in the transmit mode, 78 mA in the receive mode, and 30 mA in the sleep mode (the energy consumed to change states is not disclosed). We assume that in the NoTWT scenario, all STAs generate traffic shortly after the first beacon, then remain active for an entire beacon interval (10% of the total time in the transmit mode), while they access the channel and transmit their traffic. After that, they remain active for the following 9 beacon intervals (90% of the total time in receive state) while waiting for the next traffic generation. In the TWT scenario, instead, STAs are in transmit mode only during their assigned TWT SPs, which last about 10 ms (transmit mode for 1% of the total time). In the remaining time, they doze to save energy and avoid contending for the channel while other STAs are transmitting (99% of time in modem sleep state). Note that a voltage of 5 V is considered to calculate the power: in the ESP32-C6-DevKitC-1, the 3.3V ESP32-WROOM-1 power supply voltage is provided by a voltage linear regulator powered by the USB 5V rail. In summary, in the traffic scenario described in Sec. VII-B, we can compute the STAs' average power consumption as:

$$P_{\text{NoTWT}} = 10\% \cdot P^{\text{tx}} + 90\% \cdot P^{\text{rx}} = 0.48\text{W}$$

$$P_{\text{TWT}} = 1\% \cdot P^{\text{tx}} + 99\% \cdot P^{\text{sleep}} = 0.16\text{W}\,.$$

In other words, using TWT, the ESP32 STAs should be able to save 66% of power compared to the NoTWT scenario, consuming 0.16 W instead of 0.48 W on average.

We then compare the estimated values of average power consumption with those measured in the testbed, as presented in Tab. IV. We took these measurements with the RCE PM500 power meter connected to the USB hub powering the 10 ESP32 boards. In Tab. IV, we also report the approximate per-STA consumption, computed by subtracting the power loss of

TABLE IV
AVERAGE TESTBED POWER CONSUMPTION

Scenario	Total [W]	Per STA (approx.) [W]
NoTWT	6.343	0.47
TWT	3.454	0.24

the rectifier in the power supply of the USB hub (assuming an 80% efficiency) and the power consumption of the integrated circuit(s) inside the hub itself (450 mW). Under NoTWT, we measure a 6.3 W power consumption, which translates to a STA power consumption of 0.47 W. Instead, when TWT is enabled, the total power consumption decreases to 3.4 W, equivalent to a per-STA consumption of 0.24 W, which is 49% lower than for the NoTWT case. We observe that, in the NoTWT scenario, the estimated power closely matches the measured power. Conversely, in the TWT scenario, the estimated consumption is 33% lower than the measured one. This discrepancy is due to factors such as energy consumption while transitioning from an operational mode to another, which is not accounted for in our estimations, and modem wake-ups outside the scheduled SPs for clock re-synchronization. Nonetheless, the results confirm that TWT brings substantial power savings to Wi-Fi 6 networks, deployed with off-the-shelf devices, supporting time-sensitive applications.

C. TASPER vs. ShortestFirst: Experimental setup and results

We now focus on the experimental comparison between TASPER (with β=1) and ShortestFirst, that is, the best-performing alternative out of those we considered in Sec. V-C. The testbed configuration is the same as for the first case study in Sec. VII-B, but we now consider the TASP instance represented in Fig. 12(a), which is more representative of a real-world scenario where stations have different traffic patterns. This schedule comprises STAs that generate periodic traffic with short transmission durations and loose AoI deadlines (STAs m=1,3,4,6,7,8,9,10) and two STAs generating a long transmission, one with a tight AoI deadline (STA m=2) and the other with a loose one (STA m=5). For each STA, we set a TX priority $p_j = 10 - m$ and map the TXOP duration τ_j to a packet size of 1,272 bytes which, using MCS=0, corresponds to a number of frames of $\frac{2}{5}\tau_j$.

The results are illustrated in Figures 12(b) and (c), which show the AoI measured for each scheduled STA with the TASPER and the ShortestFirst strategy, respectively. TASPER schedules all STAs in the same order as in the problem instance representation of Fig. 12(b), with no unused air time between subsequent SPs. All the STAs always end their transmissions within the corresponding AoI deadlines, except for the 10th STA, whose transmissions exceed the AoI deadline 5.8% of the time. On average, we measure an AoI of 22.05 ms. Interestingly, we observe that on rare occasions the first STA ends its TX in less than 1 ms. An imprecise clock synchronization between the Wi-Fi TSF and the board clock, allowing the STA to start the transmission earlier, causes this behavior. Indeed, we are not employing triggered-enable TWT to signal STAs when their SPs start. In contrast, ShortestFirst first schedules STA 1 and 3, which have shorter transmission durations, leaving no room to meet the AoI deadline for the traffic of STA 2. Since it cannot schedule the TX from STA 2, it has more time to schedule the requesting STAs, yielding an average AoI of 11.66 ms (i.e., 47% lower than TASPER's). However, it fails to schedule STA 2, whose service request is thus dropped.

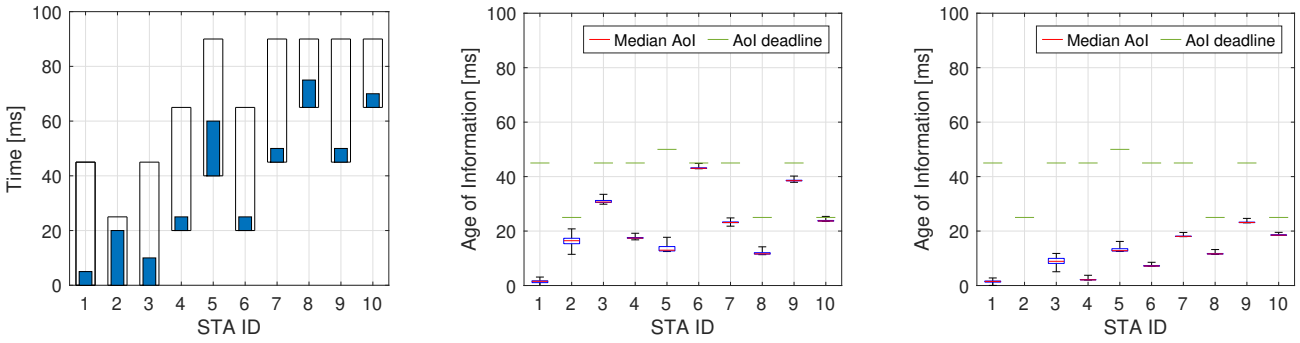
These results confirm that TASPER successfully leverages TWT to schedule time-sensitive traffic of STAs. Notably, using TWT-compatible STAs, broadly available in the market, we demonstrate TASPER's capability of admitting more transmissions than alternative approaches such as ShortestFirst by maximizing airtime utilization and avoiding AoI deadline violations.

VIII. RELATED WORK

Compared to the previous amendment (i.e., IEEE 802.11ac), IEEE 802.11ax (a.k.a. Wi-Fi 6 [3]), offers new features such as Orthogonal Frequency Division Multiple Access (OFDMA), uplink Multi-user MIMO (MU-MIMO), and TWT [15]. TWT, refined in IEEE 802.11ax but originally proposed in IEEE 802.11ah, provides STAs with a new mechanism that allows them to agree on their active and doze times. This reduces energy consumption and network delays, as it decreases medium access contention [16]. Although the standard defines the TWT mechanism, it does not provide any rules or criteria for agreement settings. Finding the optimal agreement is thus an open issue, which can be tackled by formulating a scheduling problem that aims at allocating resources (or machines) to jobs over time periods to optimize one or more objectives.

The problem of traffic scheduling is not exclusive to wireless networks such as Wi-Fi; it has also been extensively studied in Ethernet-based networks through the Time-Sensitive Networking (TSN) framework. In particular, the Time-Aware Shaper (TAS) is a TSN mechanism that enables deterministic transmission by controlling gate operations on egress queues of network nodes according to a global time schedule, thus minimizing queuing delays for time-triggered traffic. A comprehensive review of scheduling algorithms designed for TAS is presented in [17].

Extending TSN concepts to wireless networks introduces additional challenges due to the shared and time-varying nature of the wireless medium. In this context, IEEE 802.15.4 [18] and LoRa [19] have emerged as two prominent communication technologies enabling low-power wireless networking in industrial and IoT scenarios [20]–[23]. IEEE 802.15.4 provides a low-rate, short-range PHY with support for energy-efficient operation and simple MAC mechanisms such as CSMA/CA. LoRa enables ultra-long-range communication with extremely low power consumption, but at the cost of low data rates and long transmission times, making it suitable for sparse, delay-tolerant applications. Building on these technologies, a number of industrial protocols and research efforts have aimed to introduce more predictable and time-sensitive behavior. Notably, WirelessHART [12], [13], [21] extends IEEE 802.15.4 by employing centralized TDMA-based scheduling and frequency hopping to achieve deterministic and reliable communication in mesh topologies, especially in process automation and industrial monitoring. RTLoRa [24], instead, modifies Lo-



(a) Problem instance. White boxes denote the period between TX generations and AoI deadlines; blue bars indicate the TX duration.

(b) TASPER: All STAs are scheduled; for each of them, the measured AoI is reported.

(c) ShortestFirst: The AoI is reported for each STA, except for the 2nd, since it cannot be scheduled.

Fig. 12. Problem instance and experimental results: On each box of the box plots, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the 5th and the 95th percentiles. Light green segments denote the AoI deadlines.

RaWAN's purely asynchronous MAC by introducing slotted ALOHA access and lightweight scheduling primitives, with the goal of reducing collisions and supporting soft real-time traffic patterns in dense deployments, which makes it suitable for industrial use cases. While both WirelessHART and RT-LoRa represent significant advances in bringing determinism and efficiency to low-power wireless networks, they operate under fundamentally different design assumptions compared to Wi-Fi-based solutions. WirelessHART works over a low-rate PHY, relying over fixed time-slot communications, multi-hop mesh routing, and channel hopping. RTLoRa, instead, targets long-range communication scenarios, trading off latency and throughput for energy efficiency and coverage, and is constrained by strict duty-cycle regulations and limited bandwidth. In contrast, Wi-Fi networks such as those targeted in this work rely on single-hop communication, and are designed to support high data rates and highly dynamic traffic patterns.

Concerning Wi-Fi based networks, [31] gives an in-depth overview of the IEEE 802.11 standards evolution to support deterministic communication. IEEE 802.1Qbv for time-aware queue gating, IEEE 802.1CB for frame replication and elimination, and AP-driven OFDMA scheduling, can be leveraged to reduce contention and enhance predictability without relying on restrictive TWT operations.

Building upon this, scheduling strategies based on TWT have been investigated by several works. Using broadcast TWT, the scheduler in [25], based on a genetic algorithm, takes advantage of OFDMA to avoid channel contention: by allocating dedicated resources to each STA, no more than one STA is active at any time. Differently from our work, rather than directly allocating SPs, [25] uses TWT to wake up a number of STAs that is lower or equal to the number of disjoint sets of tones (Resource Units, or RUs), so each STA gets its own RU without colliding with others. Also using OFDMA, [30] proposes a traffic-awareness-based TWT scheduling scheme that leverages spatio-temporal traffic prediction and classification to dynamically adjust TWT parameters and optimize resource allocation across time and frequency. Their scheme, leveraging a greedy scheduling algorithm, improves energy efficiency and QoS by tailoring wake-up intervals and durations to predicted traffic patterns. [26], instead, focuses on throughput and fairness, and proposes two TWT schedulers: a max-rate scheduler, which aims to maximize the overall network throughput, and a proportional fair scheduler, which tries to balance network throughput and fairness so that a minimal level of service is guaranteed to all users. Aiming at higher energy efficiency and uplink throughput, [28] presents a power-saving scheme for overlapping BSS. However, such a scheme schedules TWT SPs of the same duration, thus wasting radio resources in case of short transmissions. Still considering high-density scenarios with overlapping BSS, [29] introduces a TWT-based AP coordination scheme to avoid interference. Here, the key idea is to allocate interfering STAs to different TWT SPs, and to give priority to STAs with high traffic volumes, which leads to increased throughput and energy efficiency.

As for TWT applied to IIoT scenarios specifically, [27] extends a wired Time Sensitive Networking (TSN) network to the wireless domain, employing broadcast TWT to separate traffic flows according to their priority and then different RUs to isolate each flow. However, experimental results show poor performance, as the TWT SPs are emulated through WoWLAN instead of being natively implemented by the used transceivers. Looking instead at the performance on Commercial Off-the-Shelf (COTS) devices, [32] investigates the performance of TWT, in the case where TWT agreements are statically configured by STAs (Android smartphones) to decrease energy consumption and are not tailored to traffic characteristics. To the best of our knowledge, we are the first to report a performance analysis of TWT on COTS devices where the TWT SPs are dynamically scheduled based on the expected traffic patterns. The imperative to maintain data freshness has spurred significant research on AoI. Various studies have investigated AoI minimization considering aspects such as maximum AoI thresholds and throughput constraints [33], [34]. However, these AoI-centric formulations typically do not incorporate energy consumption as a primary optimization objective or scheduling constraint, which is a core element of our work where TWT is central.

Finally, a preliminary version of our work has appeared

TABLE V
COMPARISON OF TASPER WITH RELATED WORKS ON WIRELESS INDUSTRIAL NETWORKS AND TWT SCHEDULING STRATEGIES

Work (Year) [Ref]	Optimization Objective	Optimization Technique	PHY Layer	Real-Time Guarantee	Energy-Aware	AoI Support	Evaluation Method
TASPER [Ours]	Maximize traffic acceptance and energy efficiency under AoI constraints	MILP model (TASP) solved via heuristic (TASPER)	IEEE 802.11ax	Soft (AoI-based)	Yes	Yes	ns-3 simulation and real testbed
Chen et al. (2018) [12]	Minimize weighted end-to-end delay	Iterative hop-wise scheduling algorithm (HSA) based on MWIS	IEEE 802.15.4	Soft	No	No	MATLAB simulation
RT-LoRa by Leonardi et al. (2019) [24]	Provide bounded end-to-end delay	Centralized, superframe-based TDMA strategy	LoRa	Hard	Yes (via QoS classes)	No	OMNeT++/FLoRa simulation
Chen et al. (2021) [25]	Maximize throughput under delay constraints	Genetic algorithm	IEEE 802.11ax	Soft (delay-bound)	Yes	No	Simulation with traffic generator
Yang et al. (2021) [26]	Maximize uplink throughput and fairness for TWT STAs	Heuristic grouping: max-rate and proportional-fair algorithms	IEEE 802.11ax	None	No	No	ns-3 simulation (OFDMA uplink)
Schneider et al. (2022) [27]	Minimize latency and jitter for TSN traffic	Time-aware scheduling with TWT SP alignment to TAS	IEEE 802.11ax	Soft (latency/jitter bounds)	No	No	Real hardware testbed
Chen et al. (2022) [28]	Maximize energy efficiency under delay bounds in OBSS	Graph-coloring grouping + deterministic algorithm	IEEE 802.11ax	Soft (delay-bound)	Yes	No	Simulation with traffic generator
Peng et al. (2024) [29]	Minimize fixed-duration TWT SPs usage under throughput constraints	Greedy algorithm	IEEE 802.11ax	None	Yes	No	Custom simulator
Dang et al. (2024) [30]	Improve channel efficiency and reduce power consumption under dynamic traffic	Deep learning traffic prediction + heuristic TFST scheduler	IEEE 802.11ax	Soft (service-specific)	Yes	No	Custom simulator with traffic model

in [35], where however we introduced a simpler version of the TASP problem and just sketched the TASPER algorithm, while showing its performance obtained via simulation only and under a smaller-scale scenario.

Novelty. In summary, the innovative features of our work are two-fold. First, concerning energy-saving scheduling, ours is the first work on time-sensitive traffic scheduling that addresses energy-saving besides trying to maximize the number of scheduled transmissions. Second, regarding AoI requirements, existing scheduling problems are designed to comply with delay requirements and do not account for information freshness. Instead, our problem considers AoI constraints to guarantee that the delivered data is not outdated and provides new, fresh information.

IX. CONCLUSIONS

In this paper, we presented a novel solution for efficient TWT scheduling in time-critical IIoT scenarios. First, we provided a set of motivational findings, showing the advantages of TWT in guaranteeing low and deterministic latency as well as in saving energy. We then proposed a mathematical model of a TWT-enabled Wi-Fi network and formulated the TWT Acceptance and Scheduling Problem (TASP), proving its NP-hardness. In light of the problem complexity, we envisioned an efficient heuristic algorithm, named TASPER, and demonstrated its effectiveness with respect to baseline strategies. Numerical results, obtained using a realistic IIoT scenario and through our ns-3-based TWT simulator, show that TASPER achieves a 24.97% lower mean rejection cost, and up to 14.86% lower energy consumption than the best performing baseline. Additionally, TASPER outperforms HSA, a state-of-the-art solution adapted from WirelessHART [12], reducing the mean rejection cost by up to 26%, and the energy consumption by up to 34%. Finally, using our IIoT TWT-compatible testbed, we validated TASPER's effectiveness as a traffic scheduling strategy, demonstrating that it admits more transmissions than simpler alternatives without incurring any AoI deadline violations.

Future work will extend TASPER to an OFDMA scenario, where Wi-Fi transmissions are allocated in Resource Units spanning both the time and frequency dimensions, and further investigate the stations' energy consumption.

REFERENCES

[1] S. Munirathinam, "Industry 4.0: Industrial Internet of Things (IIoT)," in *Advances in computers*. Elsevier, 2020, vol. 117, no. 1, pp. 129–164.

[2] S. Krishnan Venkateswaran, C.-L. Tai, R. Garnayak, Y. Ben-Yehezkel, Y. Alpert, and R. Sivakumar, "IEEE 802.11ax Target Wake Time: Design and Performance Analysis in ns-3," in *2024 Workshop on ns-3 (WNS3 2024)*, 2024, pp. 1–9.

[3] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN," *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–767, 2021.

[4] ns-3 community. (2024) ns-3-dev Wi-Fi Module Design Documentation. https://www.nsnam.org/docs/models/html/wifi-design.html.

[5] C. Oguz, F. Sibel Salman, and Z. Bilgintürk Yalçın, "Order acceptance and scheduling decisions in make-to-order systems," *International Journal of Production Economics*, vol. 125, no. 1, pp. 200–211, 2010.

[6] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey," in *Discrete Optimization II*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, and B. Korte, Eds. Elsevier, 1979, vol. 5, pp. 287–326. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016750600870356X

[7] M. de Weerdt, R. Baart, and L. He, "Single-machine scheduling with release times, deadlines, setup times, and rejection," *European Journal of Operational Research*, vol. 291, no. 2, pp. 629–639, 2021.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720308468
[8] J. Chuzhoy and R. Ostrovsky, "Approximation algorithms for the job interval selection problem and related scheduling problems," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 348–356.
[9] S. Burer and A. Saxena, "The MILP road to MIQCP," *Mixed integer nonlinear programming*, pp. 373–405, 2011.
[10] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.
[11] F. Raviglione, C. R. Carletti, M. Malinverno, C. Casetti, and C. Chiasserini, "ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services," *Computer Communications*, vol. 217, pp. 70–86, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424000227
[12] G. Chen, X. Cao, L. Liu, C. Sun, and Y. Cheng, "Joint scheduling and channel allocation for end-to-end delay minimization in industrial WirelessHART networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2829–2842, 2018.
[13] International Electrotechnical Commission, "IEC 62591:2016 - Industrial communication networks – Wireless communication network and communication profiles – WirelessHART," Available from IEC Webstore, 2016, standard.
[14] E. Systems. (2024) ESP32-C6-WROOM-1 & WROOM-1U Datasheet v1.1. https://www.espressif.com/sites/default/files/documentation/esp32-c6-wroom-1_wroom-1u_datasheet_en.pdf. Accessed: 2024-11-21.
[15] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A Tutorial on IEEE 802.11ax High Efficiency WLANs," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 197–216, 2019.
[16] M. Nurchis and B. Bellalta, "Target Wake Time: Scheduled Access in IEEE 802.11ax WLANs," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 142–150, 2019.
[17] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-Sensitive Networking (TSN)," *IEEE Access*, vol. 11, pp. 61 192–61 233, 2023.
[18] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020.
[19] Semtech Corporation, "LoRa® and LoRaWAN®: A Technical Overview," https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf, feb 2020.
[20] F. Chen, T. Talanis, R. German, and F. Dressler, "Real-time enabled IEEE 802.15. 4 sensor networks in industrial automation," in *2009 IEEE International Symposium on Industrial Embedded Systems*. IEEE, 2009, pp. 136–139.
[21] D. Chen, M. Nixon, S. Han, A. K. Mok, and X. Zhu, "WirelessHART and IEEE 802.15.4e," in *2014 IEEE International conference on industrial technology (ICIT)*. IEEE, 2014, pp. 760–765.
[22] M. Luvisotto, F. Tramarin, L. Vangelista, and S. Vitturi, "On the use of LoRaWAN for indoor industrial IoT applications," *Wireless Communications and Mobile Computing*, vol. 2018, no. 1, p. 3982646, 2018.
[23] H. H. R. Sherazi, L. A. Grieco, M. A. Imran, and G. Boggia, "Energy-Efficient LoRaWAN for Industry 4.0 Applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 891–902, 2021.
[24] L. Leonardi, F. Battaglia, and L. Lo Bello, "RT-LoRa: A medium access strategy to support real-time flows over LoRa-based networks for industrial IoT applications," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 812–10 823, 2019.
[25] Q. Chen and Y.-H. Zhu, "Scheduling Channel Access Based on Target Wake Time Mechanism in 802.11ax WLANs," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1529–1543, 2021.
[26] C. Yang, J. Lee, and S. Bahk, "Target Wake Time Scheduling Strategies for Uplink Transmission in IEEE 802.11ax Networks," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.
[27] B. Schneider, R. C. Sofia, and M. Kovatsch, "A Proposal for Time-Aware Scheduling in Wireless Industrial IoT Environments," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–6.
[28] Q. Chen, "An Energy-Efficient Channel Access With Target Wake Time Scheduling for Overlapping 802.11ax Basic Service Sets," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 973–18 986, 2022.
[29] X. Peng, Y. Fang, C. Li, and L. Guo, "Access Point Coordination Based TWT Scheduling for the Next Generation WLAN," in *2024 13th International Conference on Communications, Circuits and Systems (ICCCAS)*, 2024, pp. 238–243.
[30] Z. Dang, S. Yan, X. Gu, and Y. Chang, "Traffic Awareness-Based Target Wake Time Scheduling in 802.11ax WLANs," *International Journal of High Speed Electronics and Systems*, vol. 34, no. 01, p. 2540080, 2025.
[31] D. Cavalcanti, C. Cordeiro, M. Smith, and A. Regev, "WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11," *IEEE Communications Standards Magazine*, vol. 6, no. 4, pp. 22–29, 2022.
[32] C. Zhao, B. Li, S. Wang, and T. He, "The First Measurement Study of Target Wake Time Mechanism in 802.11ax on COTS Devices," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 4695–4700.
[33] C. Li, Q. Liu, S. Li, Y. Chen, Y. T. Hou, W. Lou, and S. Kompella, "Scheduling With Age of Information Guarantee," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2046–2059, 2022.
[34] I. Kadota, A. Sinha, and E. Modiano, "Scheduling Algorithms for Optimizing Age of Information in Wireless Networks With Throughput Constraints," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1359–1372, 2019.
[35] C. Puligheddu, F. Busacca, R. Rusca, F. Raviglione, C. Casetti, C. F. Chiasserini, and S. Palazzo, "Target Wake Time Scheduling for Time-Sensitive Networking in the Industrial IoT," in *2024 IEEE 35th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2024.

Fabio Busacca is an assistant professor at the University of Catania, Italy. His main research interests are LPWAN protocols for the IoT, AI applied to next-generation communication networks, and underwater networks.

Corrado Puligheddu is an assistant professor at Politecnico di Torino, Italy. His main area of interest is the application of machine learning to wireless networks, focusing on radio resource management and network orchestration.

Francesco Raviglione is an assistant professor at Politecnico di Torino, Italy. His main areas of interest are wireless and vehicular networks.

Riccardo Rusca is a research fellow at Politecnico di Torino, Italy. His main areas of interest are crowd monitoring and time sensitive networking.

Claudio Casetti is a Full Professor with Politecnico di Torino, Italy. His research interests are vehicular networks, ITS, 5G/6G, and IoT systems.

Carla Fabiana Chiasserini is Full Professor with Politecnico di Torino, Italy. Her research interests are in the design, modeling, and performance evaluation of mobile networks and services.

Sergio Palazzo is a Full Professor with the Università di Catania, Italy. His research interests include mobile systems, wireless and satellite networks, and traffic engineering.