Target Wake Time Scheduling for Time-Sensitive Networking in the Industrial IoT

Corrado Puligheddu*, Fabio Busacca†, Riccardo Rusca*, Francesco Raviglione*,
Claudio Casetti*, Carla Fabiana Chiasserini*, Sergio Palazzo†
* Politecnico di Torino, Italy; † University of Catania, Italy; ‡ Chalmers University of Technology, Sweden

***Abstract*—Time Sensitive Networking (TSN) is fundamental for the low-latency, reliable, and energy-efficient 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 using Target Wake Time (TWT) to instruct Wi-Fi stations to wake up and transmit in non-overlapped TWT Service Periods (SPs) and sleep in the remaining time. In this paper, we first formulate the TWT Acceptance and Scheduling Problem (TASP), whose objective is to schedule TWT SPs as to maximize traffic throughput and energy efficiency while respecting Age of Information (AoI) constraints. Then, since the TASP is NP-hard, we propose the TASP Efficient Resolver (TASPER), a heuristic strategy to find near-optimal solutions efficiently. Finally, we compare TASPER with several baselines through numerical analysis and simulations, which we performed using a TWT-compatible simulator based on ns-3. We demonstrate that TASPER schedules traffic with up to 21.23% higher priority-weighted admission ratio and saves up to 7.42% energy compared to the ShortestFirst strategy, all while satisfying AoI constraints for 99.5% of transmissions.**

***Index Terms*—target wake time, scheduling, time-sensitive networking, IIoT**

I. Introduction

The recent rise to prominence of new technological paradigms, including the Industrial Internet of Things (IIoT), Artificial Intelligence (AI), and cyber-physical systems, has radically changed the industrial domain, paving the way for the so-called Industry 4.0 [1]. One of the key innovations of Industry 4.0 is the concept of *smart factories*, where interconnected systems, data-driven decision-making, real-time data analytics, energy efficiency, and production automation redefine traditional manufacturing. Specifically, the usage of IIoT-based sensor and actuator devices is envisioned to supervise the whole industrial process, by controlling stations in production lines and detecting possible faults and anomalies. The timely and reliable communication of such devices is fundamental to prevent costly interruptions to the production chain, and, therefore, avoid substantial monetary losses. However, the shared nature of the wireless channel could be the source of unacceptable delays, as interference and collisions from concurrent devices may undermine the successful transmission of alert messages. While the usage of a retransmission mechanism could help in ensuring the delivery of the message, this may still have an impact on the so-called *Age of Information* (AoI), i.e., the time elapsed since the generation of the information until it is received by the intended recipient. Indeed, a message with a high AoI might get too old to be useful, e.g., if a production anomaly is reported too late, it may leave insufficient time for effective countermeasures, potentially leaving a production chain stoppage as the sole solution. To overcome this issue, dedicated wireless channels could be assigned to each IIoT device. However, this solution is rarely feasible due to the sheer scale of IIoT networks, which may consist of hundreds, if not thousands, of devices sharing limited spectrum resources. Hence, the only viable possibility is to schedule the utilization of shared spectrum resources accurately. In such a perspective, Target Wake Time (TWT) is a perfect fit for time-sensitive IIoT. TWT is a feature introduced in Wi-Fi 6 (IEEE 802.11ax) that allows an Access Point (AP) to schedule when each station (STA) in the network should wake up to transmit and receive data. Such a feature brings several advantages:

- **Energy saving**: IIoT devices may be battery-powered, thus having limited energy capacity. By allowing devices to mostly sleep and wake up only during specific scheduled times to exchange traffic, they can save substantial energy and improve battery life;
- **Channel contention mitigation**: by coordinating wake times, STAs can take turns accessing the channel, reducing the likelihood of collisions and contention. This noticeably improves the determinism in the communication, when dealing with safety-critical scenarios, and improves communication latency, thus reducing the AoI.

Overall, TWT represents a key enabler for latency-sensitive IIoT networks. However, while it defines *how* to manage the synchronization mechanism, it does not state *when* to wake up STAs. Hence, this calls for the design of an efficient scheduling mechanism to support TWT-based spectrum allocation in IEEE 802.11ax IIoT networks. Such a solution should guarantee a maximum AoI to STA messages while maximizing the energy efficiency of the network. Indeed, a device should not wake up too early to save as much energy as possible, nor too late, to ensure the timely delivery of packets.

Given the timeliness and importance of the problem, we propose the TWT Acceptance and Scheduling Problem Effi-

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.

cient Resolver (TASPER), an algorithmic solution for energy-efficient, AoI-constrained TWT scheduling in Wi-Fi networks. Our contributions can be summarized as follows:

- We devise a novel model for TWT-enabled IIoT 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 (i.e., not scheduled) traffic and energy consumption while guaranteeing the satisfaction of the maximum AoI constraint. Finding a solution to the TASP comprises the choice of *(i)* whether to admit the traffic and *(ii)* the target wake time(s) of each STA in the network. The possibility of rejecting traffic allows 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, which builds upon the BALAS algorithm in [2]. Our approach is specifically designed to account also for energy consumption and to find a solution in an amount of time comparable with the inter-beacon interval of the Wi-Fi networks;
- We evaluate our scheme, leveraging a realistic TWT-enabled 802.11ax simulator, and show that it offers substantial improvements over baseline strategies. Specifically, TASPER obtains up to 21.23% higher priority-weighted admission ratio and saves up to 7.42% energy compared to the best baseline, i.e., ShortestFirst.

II. Related Work

IEEE 802.11ax, commercially known as Wi-Fi 6, has been standardized in 2021 [3]. Compared to the previous amendment, IEEE 802.11ac, it offers new features such as Orthogonal Frequency Division Multiple Access (OFDMA), uplink Multi-user MIMO (MU-MIMO), and TWT [4]. TWT, which has been refined in IEEE 802.11ax but originally proposed in IEEE 802.11ah, provides stations with a new mechanism that allows them to agree on their active and doze times. This reduces energy consumption and improves network delays thanks to the avoidance of medium access contentions [5]. The standard defines the TWT mechanism but provides no rule or criteria for agreement settings, leaving the implementation details to equipment vendors. Finding the optimal agreement is essentially a scheduling problem, a class of problems that deals with the allocation of resources (or machines) to jobs over time periods to optimize one or more objectives.

Several works have investigated scheduling strategies based on TWT. In [6], two TWT schedulers are proposed: a max-rate scheduler, which aims to maximize the overall network throughput, and a proportional fair scheduler, which instead tries to best balance network throughput and fairness so that a minimal level of service is guaranteed to all users. They use the ns-3 simulator to evaluate the performance and show the benefits in terms of throughput and fairness. In [7], a power save scheme for higher energy efficiency and better throughput for UL traffic is proposed for overlapping basic service set (BSS). However, such a scheme schedules TWT windows of the same duration, thus wasting radio resources in case of short transmissions. As for TWT applied in the IIoT, [8] tries to extend a wired Time Sensitive Networking (TSN) network to the wireless domain, employing TWT to isolate concurrent traffic according to the priority and showing that this approach leads to lower latency for high-priority traffic.

Novelty. The innovative features that differentiate our work from the state of the art are two-folded:

- **Energy-Saving Scheduling**: to the best of our knowledge, this is the first work on time-sensitive traffic scheduling that addresses energy-saving besides trying to maximize the number of scheduled transmissions.
- **AoI requirements**: existing scheduling problems are designed to comply with latency 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.

III. TWT operation and motivating findings

Target Wake Time allows STAs to negotiate awake periods with the AP to exchange traffic and to enter a doze mode during the remaining time to save energy. The awake period, also called Service Period (SP), is the time period in which a station is awake to receive or send data. A TWT session, which can be composed of one or more periodic SPs, requires a preliminary TWT agreement between the TWT requesting STA and TWT responding STA. The parameters and features that are negotiated for the agreement are:

- Target Wake Time: time instant in which the STAs should wake up (if they are not already) for the TWT session;
- TWT Wake Interval: time interval between subsequent SPs for the STA (0 if the session is explicit);
- Minimum TWT wake duration: minimum time duration STAs should stay active since the beginning of the SP;
- TWT Channel: usage of subchannel selective transmission (SST) to exchange traffic in a SP;
- TWT Protection: usage of mechanisms to protect a SP from transmission of other STAs, such as Network Allocation Vector (NAV) protection.

Moreover, a TWT session can further be characterized as:

- Individual, if only a couple of STAs are involved; Broadcast, if the AP schedules a group of STAs together;
- Explicit, if it requires a new agreement to schedule another SP; Implicit instead if SPs are periodic;
- Trigger-enabled, when the AP uses Trigger frames to schedule STA transmissions;
- Announced, in which STAs send a dedicated frame (e.g., PS-Poll) when the SP starts, to signal that they are awake;
- Flexible, when the next TWT of a periodic session can be changed without requiring a new agreement.

Besides optimizing energy consumption, TWT can also be employed to provide deterministic latency to devices communicating over a shared wireless medium. However, the contention

and collision resolution mechanisms employed in Wi-Fi introduce randomness. Consequently, the timing of channel access and successful data transmissions is not deterministic. This leads to an unacceptable level of unpredictability, especially for time-critical Industry 4.0 use cases. This concept is demonstrated in Fig. 1, which shows the average latency of 8 STAs with TWT enabled and disabled, in a scenario simulated by our ns-3-based solution (details in Sec. VI), and as the payload size is increased. All STAs generate traffic at the same time, and the TWT approach schedules non-overlapped windows for each STA. TWT lets the user maintain a latency almost equal to the one the devices would experience without any contention, with low jitter and higher determinism (the confidence intervals are indeed smaller). Instead, if TWT is disabled and contention occurs, the latency is much higher, and its variation is significantly more evident.

TWT alone does not provide any guaranteed AoI, as Wi-Fi frames could wait in the transmission queue, before they are scheduled in a TWT window, for more than what the maximum tolerable AoI allows. For this reason, it is imperative to use a scheduling strategy aware of transmission buffer generation times and deadlines. This behavior is presented in Fig. 2, which introduces in the top diagram an example of a TWT acceptance and scheduling problem. For a scheduling solution to be feasible, the transmission (black bars) cannot overlap in time, as they would cause channel contention. Instead, they have to be scheduled sequentially within the grey bars, which indicate the ranges of transmission that respect the deadlines. If a naive FIFO approach is used (middle diagram), only a few transmissions can be scheduled, as other transmissions are canceled because they miss the deadlines. If instead the optimal strategy for scheduling the highest number of transmissions is used (bottom diagram), more transmissions can be accommodated (twice in this specific case). Note that either approaches are aware of deadlines, which prevents transmissions that would violate them.

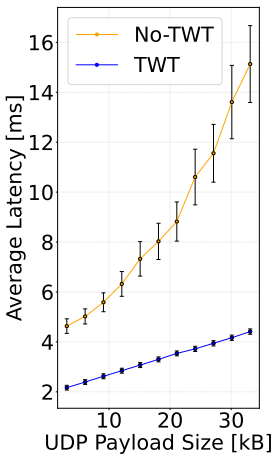


Fig. 1: Comparison of the average STA transmission latency measured with and without TWT, when used to avoid channel contention. Error bars show the 95% confidence intervals.

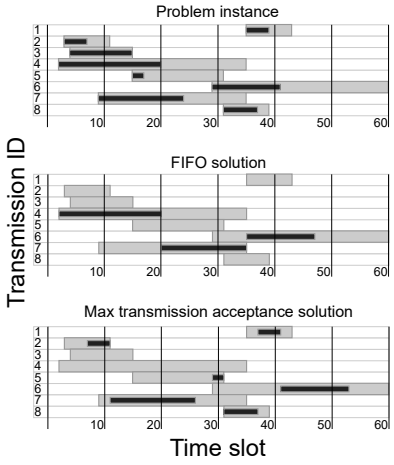


Fig. 2: TASP example (top) and solutions using a FIFO strategy (middle) and the optimal strategy for the TASP (bottom). The black bars indicate the transmission times, which have to start and end inside the grey bars to satisfy the release and deadline constraints.

IV. System model

A BSS has $M+1$ IEEE 802.11ax High Efficiency (HE) STAs: a single AP STA (AP hereafter) and M non-AP STAs (STA hereafter). STAs have to send time-sensitive traffic to the AP which, to avoid non-deterministic delays due to channel contention, employs the TWT mechanism to isolate the STA transmissions in the time domain while trying to schedule the highest amount of traffic. Traffic is sent by a STA in the AP-scheduled TWT SPs. During a SP, the scheduled STA makes a single transmission (TX hereafter) of the content of its transmission buffer, which carries time-sensitive traffic generated at the application layer. A scheduling period, i.e., the time interval between consecutive scheduling decisions, is delimited by beacon frames transmitted periodically every T_b by the AP and includes the TWT scheduling decisions valid until the next beacon transmission. A beacon period is divided into D slots of duration $T_s = T_b/D$. A slot defines the temporal resolution of TWT scheduling, meaning that a TWT window starts at the beginning of a slot and lasts for a discrete number of slots. Note that, in a scheduling period, a single STA may be scheduled multiple times if more than one TX is requested. Every STA m is associated with a signal quality level l_m, which sets the maximum data rate $\rho_m(l_m)$ that can be reliably used by the STA to exchange traffic, and that varies according to the STA capabilities and the network configuration (e.g., channel bandwidth, MIMO).

The traffic that has to be scheduled by the AP is composed of n TXs. The i-th TX is characterized by *(i)* the source STA m_i, *(ii)* the buffer size b_i, that corresponds to the amount of data that m_i has to transmit to the AP, *(iii)* the release time r_i, i.e., the time the buffer generated by an application is ready to be transmitted, *(iv)* a hard AoI deadline d_i within which the TX has to be received, and *(v)* a priority v_i associated with the criticality of the information contained in the i-th TX. We assume the propagation delay to be negligible, i.e., the TX is received as soon as the sender ends its TX. Once the maximum data rate ρ_m and the byte size of the TX b_i are known, the time duration p_i of every TX can be determined through an analytical expression or experimentally. Note that, for the sake of simplicity, r_i, p_i, and d_i are all expressed in time slots.

The m-th STA is characterized by a specific transmission power P_m^{tx}. We assume P_m^{tx} to be constant regardless of the adopted data rate. Hence, the per-slot energy consumption of STA m is equal to $E_i^{tx} = P_m^{tx} \cdot T_s$ when the STA is performing the i-th TX. Therefore, the TX of transmission time p_i requires an overall energy of $E_i^{tx} \cdot p_i$. Similarly, when active, i.e., in a state ready to transmit traffic, the m-th STA requires P_m^{rx} power, thus consuming $E_i^{rx} = P_m^{rx} \cdot T_s$ every slot. To avoid waste of energy, before and after the i-th TX, STAs transition from the sleep state to the active state and vice versa. Such transitions are associated with the total energy cost E_i^{onoff},

TABLE I: Notation

Symbol	Description
$m_i \in \{1,\dots,M\}$	Wi-Fi STA performing the i-th transmission
l_m	Signal quality level of the m-th STA-AP link
$\rho_m(l_m)$	Maximum data rate for the m-th STA
$p_i(\rho_m, b_i)$	Time duration of the i-th buffer transmission
D	Number of time slots in a scheduling period
T_b	Time interval between consecutive beacon frames
T_s	Time duration of a slot, which defines time granularity
r_i	Release time for the i-th transmission
d_i	AoI deadline for i-th transmission
v_i	Priority of the i-th transmission
P_m^{tx}	Power consumed by m_i when transmitting
P_m^{rx}	Power consumed by m_i when active
E_i^{tx}	Energy consumed by m_i in a transmission time slot
E_i^{rx}	Energy consumed by m_i in a time slot while ready
E_i^{onoff}	Energy consumed by m_i by activating and deactivating
s_{ij}	Auxiliary parameter, taking 1 if $m_i = m_j$, 0 otherwise
x_i	Transmission admission binary variable
y_{ij}	Transmission sequence binary variable
z_i	Transmission completion time variable
e_i	Total energy consumed by m_i for the i-th transmission

which may not be convenient in case of very short sleep times when, instead, staying active may save energy.

Ideally, our goal is to find a schedule that can accommodate all of the TX requests, while also keeping an eye on the overall energy consumption of the system. However, in the case of several overlapping TXs with similar requirements, scheduling all of them may not be possible. In such cases, we can reject one or more TXs, e.g., those with the lowest priority, or the longest ones. The choice of *which* TXs to reject strictly depends on the trade-off between energy consumption and traffic priority. For instance, the scheduling algorithm may choose to discard long, albeit high-priority, TX to reduce energy consumption. These two objectives are weighted by the α coefficient, which sets the relative importance of the two objective components.

We consider the decision variable to be as follows:

- $\boldsymbol{x} = [x_i]$, defined as the transmission acceptance vector, where the generic element x_i is binary and indicates whether the i-th TX is admitted;
- $\boldsymbol{y} = [y_{ij}]$, defined as the transmission sequence matrix, whose element y_{ij} is binary and takes 1 when the i-th TX is scheduled right before the j-th;
- $\boldsymbol{z} = [z_i]$, defined as the completion time vector containing the integer completion time for TXs.

In this formulation, two dummy TXs 0 and $n+1$ are defined to help in modeling the sequence of transmissions. They take the first and the last position of the sequence and are characterized by release dates, transmission times, and priority equal to 0, while the deadlines are instead equal to the maximum deadline of all TXs.

According to the definitions and the constraints above, the optimization problem is formulated as follows:

TWT Acceptance and Scheduling Problem (TASP)

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \quad \sum_{i=1}^{n} \left[\alpha \cdot \Big((1-x_i) \cdot v_i \Big) + (1-\alpha) e_i x_i \right] \tag{1a}$$

s.t.

$$\sum_{j=1, j\neq i}^{n+1} y_{ij} = x_i \qquad \forall i = 0, \cdots, n \tag{1b}$$

$$\sum_{j=0, j\neq i}^{n} y_{ji} = x_i \qquad \forall i = 1, \cdots, n+1 \tag{1c}$$

$$z_i + p_j y_{ij} + d_i(y_{ij} - 1) \leq z_j \quad \forall i = 0, \cdots, n, \; \forall j = 1, \cdots, n+1, \; i \neq j \tag{1d}$$

$$(r_i + p_i) x_i \leq z_i \qquad \forall i = 1, \cdots, n+1 \tag{1e}$$

$$z_i \leq d_i x_i \qquad \forall i = 0, \cdots, n+1 \tag{1f}$$

$$e_i \leq E_i^{onoff} + p_i E_i^{tx} \qquad \forall i = 0, \cdots, n+1 \tag{1g}$$

$$e_j \geq y_{ij} \Big(s_{ij} E_j^{rx} (z_j - p_j - z_i) + (1 - s_{ij}) E_j^{onoff} \Big) + p_i E^{tx} \quad \forall i = 0, \cdots, n, \; \forall j = 1, \cdots, n+1, \; i \neq j \tag{1h}$$

$$z_0 = 0, \; z_{n+1} = \max_{i=1,\cdots,n} \{d_i\} \leq D \tag{1i}$$

$$x_0 = 1, \; x_{n+1} = 1 \tag{1j}$$

$$x_i \in \{0,1\}, \; y_{ij} \in \{0,1\} \qquad \forall i = 0, \cdots, n, \; \forall j = 1, \cdots, n+1, \; i \neq j \tag{1k}$$

The cost function (1a) is the weighted sum of the rejection cost and the energy cost over all the requested TXs. The rejection cost is given by the priority of rejected tasks, while the energy cost is determined by constraints (1g) and (1h). Constraints (1b) and (1c) force each accepted TX to precede and succeed only one other TX. Constraint (1d) makes sure that if the i-th TX precedes the j-th, the completion time of the former precedes the one of the latter summed by the transmission time p_j. Instead, if the i-th and j-th TXs are not sequential, $z_j \geq 0$. Note that (1d) ensures the elimination of cycles in the transmission sequence [9]. Constraint (1e) ensures that the completion time of an accepted TX is not less than the sum of its release date and transmission time. The deadline requirement (1f) dictates that the completion time of an accepted TX cannot exceed its deadline.

Related to the energy aspects, (1g) sets the upper limit for the energy consumption of a TX, which is equal to the sum of the energy associated with the activation and deactivation of the radio transceiver, and the energy consumed for the transmission. Conversely, (1h) sets the energy consumption lower bound, which depends on whether the current TX is performed by the same STA that performed the one that precedes (s_{ij}). If the STA is the same, instead of going to sleep and in the active state again, it stays active, saving energy and consuming proportionally to the time between the end of the

preceding TX z_i and the beginning of the current one in the $z_j - p_j$ slot. Finally, (1i) and (1j) set $\boldsymbol{x}$ and $\boldsymbol{z}$ for the dummy TXs, while (1k) forces $\boldsymbol{x}$ and $\boldsymbol{y}$ to be binary.

Using the common three-field scheduling notation of [10], the TASP can be described as $1|SC_{si,td}; r_j; \text{reject}; \bar{d}_j| \sum w_j U_j$, i.e., single-machine sequence-independent completion-dependent batch setup cost scheduling with release dates, deadlines and rejection. In this context, U_j is the unit penalty of job j and w_j is its weight, so $\sum w_j U_j$ is the weighted number of tardy (i.e., not accepted) jobs. The batch setup cost refers to the STA activation cost for one or a batch of TX, considering possible idle times between TX (completion-dependent), and the subsequent deactivation cost.

The problems closest to ours are *(i)* the Order Acceptance and Scheduling (OAS) [2], described by $1|r_j; ST_{sd}; reject; \bar{d}_j| \sum_{j \notin R} w_j T_j - \sum_{j \notin R} v_j$, as it includes a cost for job setup similar to our activation and deactivation energy, and *(ii)* the single-machine Job Interval Selection Problem (JISP) [11], described by $1|r_j; reject; \bar{d}_j| \sum U_j$, which can be considered as a special case of the OAS that only tries to maximize the number of accepted tasks. Both such problems consider AoI constraints for TXs (i.e., maximum completion times for jobs), however, they do not model the cost associated with energy consumption. To the best of our knowledge, a formulation equivalent to the TASP has never been proposed before. The TASP includes both integer and continuous variables, and one quadratic constraint (1h), therefore it belongs to the MIQCP class, whose problems are known to be NP-hard [12]. Moreover, even without considering (1h) and the energy term of (1a), such a simplified problem, which can be referred to as the single-machine Job Interval Selection Problem (JISP) (i.e., $1|r_j; \text{reject}; \bar{d}_j| \sum w_j U_j$), is demonstrated to be NP-hard [11]. The optimal solution for the TASP can be calculated using MIQCP solvers such as Gurobi and IBM CPLEX. However, as scheduling decisions are to be timely calculated on a per-beacon period basis by a possibly low-power AP, they cannot be calculated for non-trivial problem instances. Since TASP is clearly NP-Hard, we propose a heuristic algorithm to efficiently find solutions.

V. Heuristic algorithm

Given the high complexity of the TASP, we devise a heuristic strategy, named TASPER (TASP Efficient Resolver), to find solutions for the TASP in a quick and efficient way. As the TASP is closely related to the Order Acceptance and Scheduling (OAS) problem (see Sec.II), we developed TASPER by extensively modifying BALAS, a heuristic algorithm proposed in [2] to solve the OAS. As compared to the original algorithm, TASPER is tailored to specifically fit the TWT scheduling problem, with a particular focus on energy saving. This approach perfectly matches our scenario as we are interested in finding solutions that, while sub-optimal, can be derived in an amount of time as small as the inter-beacon interval of the Wi-Fi network. The basic idea behind TASPER is to treat the TWT scheduling as the best path-finding problem in a decision diagram. Here, vertex i of the diagram corresponds to the scheduling of the i-th TX at the time $t_i = z_i - p_i$. The edge between vertex j and vertex i is accordingly associated to a reward $r_i = \alpha v_i + (1-\alpha)(1-e_i)$, with v_i and e_i normalized in $[0, 1]$. In such a perspective, the goal of the TASPER algorithm is to find the best path, i.e., the one with the highest cumulative value, between the dummy TX 0 (the *source*) and the dummy TX $n+1$ (the *sink*). Hence, TX 0 is always scheduled first. Note that maximizing the cumulative reward r_i corresponds to minimizing the objective function (1a). First, the remaining TXs are ordered according to their latest start time $d_i - p_i$. According to the TASPER algorithm, the next TX can be selected only among those falling in a specific time window, the so-called *neighborhood*. The latter is defined by the $\hat{w}$ parameter, i.e., the neighborhood size. Specifically, a TX j is within the neighborhood of transmission i if its release time r_j is at most $\hat{w}$ time slots before or after the completion time of TX i, z_i. Here, the $\hat{w}$ parameter value gauges the trade-off between the algorithm complexity and the optimality gap. Indeed, the larger the $\hat{w}$, the broader the choice of the TX to be scheduled, with a clear impact on the problem complexity. Accordingly, given an already scheduled TX i, its neighborhood can be defined as $\mathcal{N}_i = \{j : |r_j - z_i| \leq \hat{w}\}$

The key idea of the TASPER is to recursively select the *dominant* TX choice in the neighborhood $\mathcal{N}_i$, that is the one that yields *(i)* the highest reward r_i, and *(ii)* the shortest completion time for the same reward. In such a way, it is possible to select the TXs with a high rejection cost, as well as a low completion time, to leave more room for the scheduling of other TXs. This allows the maximization of the overall reward and the number of accepted TXs as well. Since some basic concepts (e.g., the dominance) are in common with the BALAS algorithm, we refer the reader to [2] for further details.

Furthermore, TASPER implements an additional step, as compared to the vanilla BALAS algorithm. Specifically, once the scheduling solution is determined, TASPER attempts to shift subsequent TXs of the same STA to reduce as much as possible the time gap between the completion of one TX and the start of the next one. This is intended to chain the TXs as much as possible, avoiding continuously turning on and off the STA, to the benefit of its energy consumption.

VI. Evaluation methodology

To evaluate the performance of TASPER, we devised a testing methodology comprising (i) the generation of TASP instances, (ii) the derivation of the solutions of the TASP instances by TASPER, as well as several baseline solving strategies for comparison purposes, and (iii) the evaluation of the obtained solutions through network simulations.

Problem instances generation. To generate problem instances, we use a method that employs realistic network scenarios and avoids trivial solutions. Specifically, we consider 16 STAs and 16 TXs, each of which is randomly assigned to a STA. STAs are positioned such that their distance from the AP falls within a uniformly distributed interval. Such interval is delimited by the distances for which the maximum Modulation

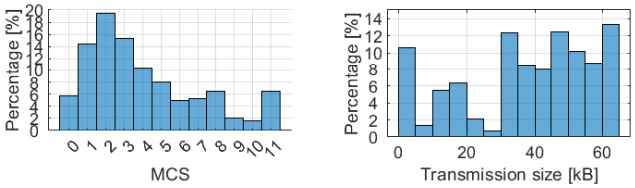


Fig. 3: Distribution of the best MCS usable for STA TXs (left) and the TX byte sizes (right) for the generated TASP instances.

and Coding Schemes (MCSs) that can be used for reliable data transmission are MCS 11 and MCS 0, respectively. The size (in bytes) of TXs is such that, given the sampled MCS, the transmission times p_i are uniformly distributed between 0 and 20 T_s. The remaining parameters of the TASP instances are generated as in [2], with parameters $\tau = 0.2$, and $R = 0.3$. Fig. 3 shows the distributions of the STAs MCS and transmission byte sizes for the generated problem instances.

Problem solutions. Solutions to the 100 generated problem instances are derived by both TASPER and the following baseline strategies used as a benchmark:

- Optimum: it is the solution calculated by Gurobi, a well-known numerical optimization solver;
- ShortestFirst: it always schedules the STA with the shortest transmission among those available;
- FIFO: it orders the transmissions according to their release time, in ascending order, and schedules the corresponding STAs in the same sequence;
- Random: it schedules randomly STAs with ready TXs.

TASPER is run with $\hat{w} = 11$, as we found this value to offer a good trade-off between solving time and optimality gap. Further, as the Random strategy is not deterministic, for each TASP instance we consider the results obtained by averaging over 100 iterations of this solving strategy.

Solution simulations. To test our TWT scheduler and compare it with other baselines, we enhanced the state-of-the-art ns-3 Wi-Fi TWT simulator in [13] with the following new features:

- Flexible Timing Configuration: users can easily specify the traffic generation time r_i of each transmission i, the TWT SP start time ($t_i = z_i - p_i$), as set by the TWT scheduling strategy, and TWT SP duration p_i.
- Deadlines support: AoI deadlines can be set as parameters to report in log files whether they have been met.
- Specialized IEEE 802.11ax Simulation Facility: a new simulation facility, including associated helpers and applications, has been developed to perform IEEE 802.11ax simulations with TWT scheduling functionalities. Additionally, our facility supports logging into CSV files.
- Support for Multiple STA Classes: the framework now supports multiple classes of STAs, each customizable with specific parameters including the number of STAs, distance from the AP, traffic type, UL/DL configuration, channel width, and MIMO configuration (up to 4x4).
- MCS and SNR Retrieval: the employed MCS and the Signal-to-Noise Ratio (SNR) can be retrieved at the application layer from the MAC and PHY (respectively) models for each received packet, through dedicated tags.
- Transparent Logging: a novel `ReportManager` module enables seamless logging functionality to gather and store latency metrics, and produce latency distributions.

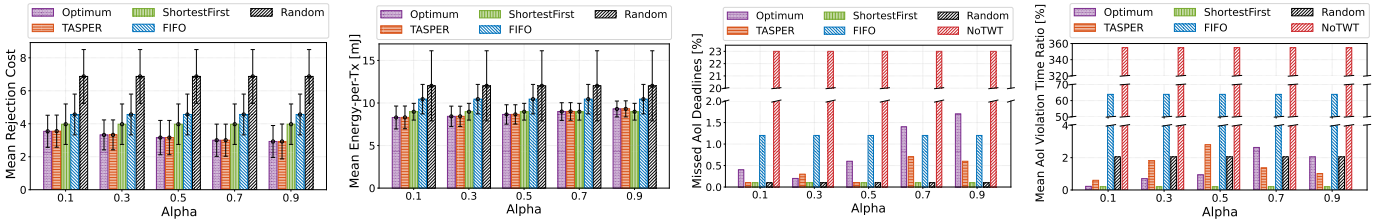
To test a realistic scenario in which TWT would bring significant advantages in terms of guaranteed deterministic latency and low energy consumption, we implemented an industrial sensor network configuration: 16 sensors (STAs) connected to an industrial AP, 2.4 GHz frequency band, SISO 20 MHz channel configuration for both the STAs and the AP, OFDMA, 800 ns Guard Interval, and up to 16 traffic transmissions from STAs. Each simulation evaluated the performance of the TWT system during one beacon period of 102.4 ms.

VII. Evaluation results

Fig. 4a reports the mean rejection cost, i.e., the sum of the priority of rejected TXs, as a function of the α weight of function (1a). Note how, as α increases, the importance of the rejection cost increases at the expense of the energy consumption. Accordingly, TASPER achieves a lower mean rejection cost for a bigger α. In all cases, not only TASPER outperforms the baselines, but also achieves near-optimal performance despite its lightweight design. Compared to the best-performing baseline, i.e., ShortestFirst, TASPER obtains up to 26% lower rejection cost, while compared to Random, such a reduction increases to 57%. In the best case, these numbers translate to a 21.23% and 193.74% higher schedule admission score (i.e., the sum of the priority of scheduled transmission) for TASPER (5.98), as compared to ShortestFirst (4.93), and Random (2.04), respectively. In the ideal case where all TXs are scheduled, the admission score is 8.91 on average.

Fig. 4b shows the per-TX energy consumption metric, computed as the ratio between the overall STA energy consumption, and the number of accepted transmissions. TASPER consumes up to 7.42% (2.56% on average) less energy than the most energy-efficient baseline, i.e., ShortestFirst, and up to 30.87% (27.24% avg.) less than Random. Clearly, as α increases, the results follow an opposite trend compared to the Mean Rejection Cost, as TASPER tends to disregard the energy consumption in favor of the rejection cost. Accordingly, $\alpha = 0.1$ yields the lowest energy consumption per completed transmission. Interestingly, for $\alpha = 0.7$ and 0.9, the ShortestFirst baseline achieves a better mean energy consumption, as it always schedules the shortest transmissions, which consume the lowest transmission energy, while TASPER schedules longer, yet more valuable transmissions requiring more energy.

Fig. 4c describes the reliability in meeting the AoI constraints, as it reports the mean percentage of missed AoI deadlines relative to all carried out TXs, derived using our custom simulator. TASPER on average violates the deadline for 0.5% of the TXs. Such violations are more frequent for higher α, i.e., when it optimizes for the rejection cost rather than the energy consumption. Curiously, Optimum yields a slight percentage of missed AoI deadlines, and even performs slightly worse than TASPER. This occurs because, as compared to TASPER, Optimum creates tighter schedules where TX completion times



(a) Mean rejection costs, i.e., the sum of the priority of rejected TX

(b) Mean energy per TX, i.e., the STA energy usage for one TX

(c) Mean percentage of missed AoI deadlines relative to all TXs

(d) Mean duration of the AoI violations relative to the TX durations

Fig. 4: Comparison of TASPER with the optimum, other baseline strategies, and a NoTWT approach as a function of the α weight coefficient, through numerical and simulation results. When present, the error bars indicate the 20th and 80th percentile.

often correspond to the deadlines. In such a way, Optimum manages to include as many TXs as possible and also to minimize the idle times between consecutive TXs from the same STA, reducing the overall energy consumption as a result. However, when problem instances include even a few TX times that deviate slightly from simulated times, meeting the scheduled completion time precisely at the deadline results in some deadlines being exceeded. Furthermore, the results are also derived for a NoTWT approach that does not isolate TXs in TWT SPs, instead, it uses the standard Wi-Fi CSMA/CA to regulate channel access. This approach is unfit for time-sensitive traffic, as it increases the number of violations by 2 orders of magnitude. The entity of the AoI violations can be observed in Fig. 4d, which shows the mean duration of the violations normalized over the TX times. When TASPER exceeds the deadline, the violations are 2% of the TX durations (0.07 ms) on average. Conversely, with a NoTWT approach, the violations last up to 3.5 times the TX durations.

VIII. Conclusions and Future Work

This paper presents a novel solution for efficient TWT scheduling in time-critical IIoT scenarios. First, we provide a set of motivational findings, showing the advantages of TWT in guaranteeing low and deterministic latencies. Then, we propose a mathematical model of a TWT-enabled Wi-Fi network and formulated the TWT Acceptance and Scheduling Problem (TASP), showing that it is NP-Hard. To solve the TASP, we present a heuristic algorithm to calculate near-optimal solutions efficiently. Finally, we report the evaluation of our solution by simulating a realistic IIoT scenario, through a customized simulator based on ns-3. Our results show the effectiveness of TASPER compared to other baseline strategies. Specifically, TASPER obtains a 21.23% higher scheduling score and up to 7.42% lower energy consumption than the best-performing baseline. Through simulation, we prove the effectiveness of TASPER and TWT in guaranteeing determinism, showing a low number of missed deadlines compared to both standard Wi-Fi networks (with no TWT) and other baseline strategies. Indeed, TASPER misses only 0.5% AoI deadlines, which is 22.9x less than the NoTWT configuration.

Future work will focus on extending the proposed scheme to an OFDMA scenario, where Wi-Fi TXs are allocated in Resource Units (RUs), spanning both the time and frequency dimensions. Then, we will perform a more extensive evaluation of energy consumption by analyzing the STA transitions between different PHY states for different scheduling policies. We also plan to investigate the application of Machine Learning in finding solutions for the TASP problem. Finally, we plan on deploying our solution on real TWT-enabled Wi-Fi 6 devices to validate our findings in the real world.

References

[1] Sathyan Munirathinam. Industry 4.0: Industrial Internet of Things (IIoT). In *Advances in computers*, volume 117, pages 129–164. Elsevier, 2020.

[2] Mathijs de Weerdt, Robert Baart, and Lei He. Single-machine scheduling with release times, deadlines, setup times, and rejection. *European Journal of Operational Research*, 291(2):629–639, 2021.

[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)*, pages 1–767, 2021.

[4] Evgeny Khorov, Anton Kiryanov, Andrey Lyakhov, and Giuseppe Bianchi. A Tutorial on IEEE 802.11ax High Efficiency WLANs. *IEEE Communications Surveys & Tutorials*, 21(1):197–216, 2019.

[5] Maddalena Nurchis and Boris Bellalta. Target Wake Time: Scheduled Access in IEEE 802.11ax WLANs. *IEEE Wireless Communications*, 26(2):142–150, 2019.

[6] Changmok Yang, Jinmyeong Lee, and Saewoong Bahk. Target Wake Time Scheduling Strategies for Uplink Transmission in IEEE 802.11ax Networks. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2021.

[7] Qinghua Chen. An Energy-Efficient Channel Access With Target Wake Time Scheduling for Overlapping 802.11ax Basic Service Sets. *IEEE Internet of Things Journal*, 9(19):18973–18986, 2022.

[8] Ben Schneider, Rute C. Sofia, and Matthias Kovatsch. A Proposal for Time-Aware Scheduling in Wireless Industrial IoT Environments. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022.

[9] Ceyda Oguz, F. Sibel Salman, and Zehra Bilgintürk Yalçın. Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics*, 125(1):200–211, 2010.

[10] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979.

[11] 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*, pages 348–356, 2001.

[12] Samuel Burer and Anureet Saxena. The MILP road to MIQCP. *Mixed integer nonlinear programming*, pages 373–405, 2011.

[13] Shyam Krishnan Venkateswaran, Ching-Lun Tai, Roshni Garnayak, Yoav Ben-Yehezkel, Yaron Alpert, and Raghupathy Sivakumar. IEEE 802.11ax Target Wake Time: Design and Performance Analysis in ns-3. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2024.