A Robust Scheduling of Cyclic Traffic for Integrated Wired and Wireless Time-Sensitive Networks

Özgür Ozan Kaynak*, Andreas Kassler*, Andreas Fischer*, Ognjen Dobrijevic†, Fabio D'Andreagiovanni‡
*Faculty of Computer Science, *Deggendorf Institute of Technology (THD)*, Deggendorf, Germany
Email: {oezguer.kaynak, andreas.kassler, andreas.fischer}@th-deg.de
†Department of Automation Technology, *ABB Corporate Research*, Vasteras, Sweden
Email: ognjen.dobrijevic@se.abb.com
‡Department of Sciences and Method for Engineering, *University of Modena and Reggio Emilia*, Reggio Emilia, Italy
Email: fabio.dandreagiovanni@unimore.it

***Abstract*—Time-Sensitive Networking (TSN) is a toolbox of technologies that enable deterministic communication over Ethernet. A key area has been TSN's time-aware traffic shaping (TAS), which supports stringent end-to-end latency and reliability requirements. Configuration of TAS requires the computation of a network-wide traffic schedule, which is particularly challenging with integrated wireless networks (e.g., 5G, Wi-Fi) due to the stochastic nature of wireless links. This paper introduces a novel method for configuring TAS, focusing on cyclic traffic patterns and jitter of wireless links. We formulate a linear program that computes a network-wide time-aware schedule, robust to wireless performance uncertainties. The given method enables robust scheduling of multiple TSN frames per transmission window using a tunable robustness parameter (Γ). To reduce computational complexity, we also propose a sequential batch-scheduling heuristic that runs in polynomial time. Our approach is evaluated by using different network topologies and wireless link characteristics, demonstrating that the heuristic can schedule 90% of 6500 requested TSN streams in a large topology.**

***Index Terms*—Time-Sensitive Networking, wireless networks and cellular networks, configuration management, time-aware traffic shaping, mathematical optimization, robust scheduling**

I. Introduction

Time-Sensitive Networking (TSN) initially started as a set of mechanisms and protocol extensions for Ethernet networks, developed by the IEEE 802.1 TSN Task Group [1]. The applicability of TSN toolbox is being extended to 5G and Wi-Fi, but with a common goal: realizing deterministic communication over integrated wired–wireless networks. TSN supports critical applications in manufacturing, aviation, and the automotive industry, which demand communication timeliness and reliability. Data is typically sent via TSN bridges/switches in streams of layer-2 frames, from talker to listener end-stations. Cyclic data, such as industrial control traffic, is sent at fixed intervals (periods). For that purpose, IEEE 802.1AS time synchronization may be used to establish a common notion of time among all communicating TSN entities. A time-aware shaper (TAS), also known as IEEE 802.1Qbv, may then be employed to schedule network traffic in TSN talkers and bridges to meet the performance requirements of the respective streams.

This work was partly funded by the Bavarian State Government through the HighTech Agenda (HTA).

A main use case for TSN is to facilitate the adoption of Industry 4.0 principles by enabling a flexible network infrastructure—one capable of accommodating the diverse requirements of industrial applications as well as advances in robotics, machine learning, and edge computing. Integrating wireless access into wired TSN networks is critical for future smart factories, which increasingly rely on mobile industrial devices. Figure 1 illustrates a motivational scenario involving cooperative task execution by autonomous mobile robots and an unmanned aerial vehicle equipped with a video camera. A TSN-based infrastructure is employed to ensure timely data delivery for closed-loop control of the cooperating robots. To achieve high communication reliability, ground robots can establish connections to the TSN backbone via 5G and Wi-Fi.

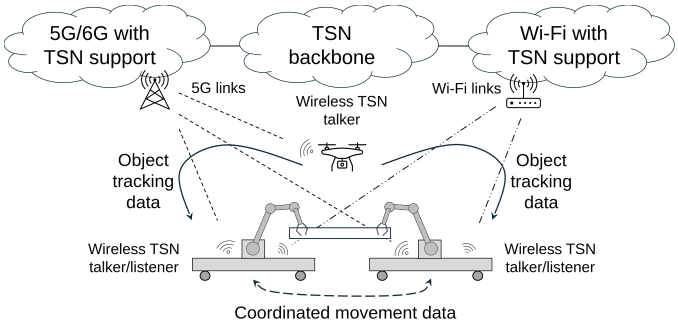

Fig. 1: TSN over 5G and Wi-Fi for cooperative industrial automation

Wireless links are inherently more susceptible to resource contention and interference than wired links, posing greater challenges for communication determinism. Interference can cause frame or acknowledgment losses, triggering layer-2 retransmissions and increasing delay variability. These variable delays can cascade through the network, potentially leading to missed delivery deadlines and impacting other streams traversing shared TSN bridges [2]. Standardization efforts have introduced features such as 5G Ultra-Reliable Low-Latency Communications (URLLC) to support deterministic and time-sensitive applications over wireless links [3]. Even with these advances, wireless segments still introduce uncertainty that complicates TAS configuration. Since a precise scheduling of gate opening and closing times, per egress port and queue, is

978-3-903176-75-1 ©2025 IFIP

required, a missed transmission window forces TSN frames to wait for the next cycle, amplifying jitter. While accounting for worst-case wireless (re)transmission delays can mitigate such an effect, it may be overly conservative, reducing efficiency or rendering schedules infeasible. Robust optimization offers a principled way to handle variability by calculating solutions feasible under data uncertainty, in turn enabling a reliable operation despite deviations from nominal values.

Addressing the aforementioned challenges of integrating wireless access into wired TSN networks, this paper focuses on scheduling cyclic traffic, as it is a dominant type in many industries. Our approach treats the wireless network as a black box, without configuring its internal working parameters (e.g., signal modulation or radio resource allocation). Instead, we assume that the wireless network designer provides a priori known lower and upper latency bounds for each wireless link, which are then used for deriving the network-wide TAS schedule. The paper contains the following contributions:

- a novel, robust integer linear problem (ILP) formulation is proposed, for configuring TAS parameters in the presence of wireless performance uncertainties (based on our previous, initial work [4]). A tunable parameter aims to balance robustness and scheduling capacity;
- a new, sequential batch-scheduling ILP heuristic is proposed, to reduce the problem complexity to polynomial time and enhance scalability to solve large problem instances fast; and
- our approach is evaluated using different network topology sizes and wireless link characteristics. We show that the exact model can only solve small problem sizes, while the given heuristic successfully schedules 90% of 6500 requested TSN streams in a large topology.

The remainder of the paper is structured as follows. Section II summarizes the state-of-the-art. The system model is described in Section III. A detailed problem formulation and the heuristic design are presented in Section IV. Section V presents evaluation results and main findings, while Section VI concludes the paper and sketches future work.

II. Related Work

The problem of scheduling cyclic traffic for TSN has been extensively studied, particularly for wired networks. Early approaches often cast the cyclic scheduling problem as a satisfiability problem [5] or an ILP [6]. Dürr *et al.* [6] propose a no-wait scheduling approach where packets are transmitted from switches immediately upon arrival at egress ports. Craciunas *et al.* [5] explore several scheduling strategies, including stream (flow) isolation, where frames of different streams are temporally isolated, and frames of the same stream are transmitted back-to-back. These foundational works typically assume deterministic link characteristics.

The integration of wireless links into TSN is an emerging research area driven by industrial needs. Zanbouri *et al.* [2] provide a comprehensive overview of the challenges and considerations when dealing with wireless uncertainties in TSN. Recent works by Egger *et al.* [7] consider a design where two wired networks are connected through a 5G logical bridge, focusing on schedulability and providing formal guarantees. Ginthör *et al.* [8] proposed a constraint programming model to configure wired schedules along with 5G resources jointly, and they extended their work by focusing on wireless signal fading [9]. Sharma *et al.* [10] model a two-stage (first routing, then scheduling) mixed ILP, considering an end-to-end wired and wireless no-wait scheduling approach. Li *et al.* [11] proposed a joint scheduling of TSN with Wi-Fi.

Robust Optimization offers a framework for handling data uncertainty in scheduling problems. Bertsimas and Sim introduced the Γ-robustness model [12] that allows users to configure system robustness via an adjustable parameter that controls the so-called "price of robustness", namely, the reduction in value that an optimal solution to the problem must face to be protected against data uncertainty. The tunability of Γ-robustness is valuable for real-world applications where uncertainties, like variable wireless delays, are prevalent. Besides the easiness of tunability, which is highly desirable from a computational point of view, we adopt Robust Optimization due to two other major advantages that it offers compared to other methods for optimization under data uncertainty, such as Stochastic Optimization: 1) embedding data uncertainty in the optimization model by defining a robust counterpart that preserves the nature of the original problem (e.g., a linear model admits a linear robust counterpart); 2) allowing to preserve the complexity of the original problem (see [12]).

Compared to previous works, our paper presents a robust ILP-based scheduling approach for integrated wired and wireless TSN that explicitly accounts for wireless delay uncertainty using a tunable robustness parameter. Additionally, we propose a scalable sequential batch-scheduling ILP heuristic that enables practical scheduling for large industrial TSNs.

III. System Modeling and Assumptions

We consider a TSN integrating wired and wireless segments (e.g., 5G, Wi-Fi). Wireless systems are treated as black boxes: their internal configuration is not modeled, but lower and upper delay bounds for each wireless link are assumed to be known (see Sec. IV-D). These bounds are used to derive TAS schedules and talker offsets for the wired TSN switches. Our approach absorbs wireless uncertainties by dimensioning buffers at the wireless bridges, controlled by a parameter.

A. Network Model

The network is modeled as a directed graph $G = (N, L)$, where nodes $n \in N$ are wired/wireless end stations or bridges, and links $l \in L$ (wired or wireless) map to egress ports $p \in P$. Scheduling focuses on Gate Control Lists (GCLs) configured by the TAS at each egress port, which control when queues can transmit. The scheduling problem determines GCLs and stream transmission offsets to meet all latency and jitter constraints [13]. Streams traverse the network via links mapped to ports; wireless stations connect through dedicated wireless bridges. All nodes are time-synchronized. For wireless bridges, per-station buffers hold frames until their

scheduled transmission time, providing functionality similar to TAS (e.g., de-jitter at a 5G node [14]).

B. Data Stream Model

A set of TSN streams $s \in S$ is requested, each following a cyclic schedule from a talker to a listener (wired or wireless). Each stream has a fixed period C_s, maximum latency $max\_latency_s$, and maximum jitter max_jitter_s. Streams traverse a subset of nodes and links, corresponding to a subset of ordered ports $P_{s,u} \subseteq P$ for a path u. The cyclic schedule allocates distinct time windows for each stream at each port along its path. We model a single egress port queue class per port; queue classes can be configured freely in post-processing.

IV. Problem Model and Solution Design

TABLE I: Modelling Notations

Sets and Indices	Description
S	Set of all data streams s.
P	Set of all egress ports p.
u	Index for a specific path of a s.
r	Index for a repetition instance of a s.
U_s	Set of candidate paths for s.
R_s	Set of repetition instances for s within the LCM_S.
$P_{s,u}$	Sequence of egress ports for s along u.
P'_p	Set of (s, u) pairs that traverse p.
Input	**Description**
C_s	Cycle period of a data stream s.
LCM_S	Least common multiple of all C_s, a greater configuration cycle time that will repeat in all devices.
$max\_latency_s$	Maximum end-to-end latency for stream s.
max_jitter_s	Maximum jitter for a stream s, talker offset variation.
Γ	Robustness level parameter, configures uncertainty delay budget, a value in $[0, 1]$.
$ft_s(k,p)$	Minimum time for the k^{th} frame of s at p.
$fd_s(k,p)$	Maximum delay deviation for the k^{th} frame of s at p.
$t(s,p)$	Lower bound time to schedule for stream s at port p.
$d(s,p)$	Delay deviation for stream s at port p.
$next\_port_{s,u,p}$	Gives the next port p' for stream s on path u after current port p.
$next(\Gamma, s, u, p, p')$	Required time offset for stream s between port p and the next port p'.
$fixed\_set(s,p)$	Function returning a set of fixed ($fixed\_start_f$, $fixed\_end_f$) time blocks on a p.
$fixed\_start_f$	Start time of a pre-scheduled, fixed block f.
$fixed\_end_f$	End time of a pre-scheduled, fixed block f.
M	A sufficiently large number for big-M constraints.
Decision Variables	**Description**
$x^{r,u}_{s,p}$	Start time of the transmission window for the r^{th} instance of s on u at p.
a_s	Binary variable: 1 if s is scheduled, 0 otherwise.
$z_{s,u}$	Binary variable: 1 if s is scheduled on u, 0 otherwise.
$y^{s',r',u'}_{s,r,u,p}$	Binary helper variable for ordering streams in temporal isolation constraints.

A. Objective

Given a set of requested streams $s \in S$, each with period C_s, the goal is to find a schedule for all ports $p \in P$. The schedule defines the path for each stream (as a sequence of ports), the talker offsets, and the GCL time windows for bridges.

B. Model Inputs and Scheduling Design

Times are represented as integers corresponding to TSN clock ticks. For each stream s, we compute up to k candidate paths using Yen's k-shortest path algorithm [15]. Each path is a sequence of ports $P_{s,u}$ to be scheduled. The GCL configuration cycle is set to the least common multiple (LCM_S) of all stream periods C_s, with $|R_s| = LCM_S/C_s$ stream instances per cycle.

Let m be the number of frames a talker transmits for a stream within one cycle. For each frame k at port p, the minimum transmission time is:

$$ft_s(k,p) = trans_{k,p} + prop_{k,p} + proc_{k,p} \tag{1}$$

For wireless links, $ft_s(k,p)$ is equal to the best-case scenario (minimum delay). Wireless links introduce delay uncertainty, modeled by $fd_s(k,p)$ (delay deviation); for wired links, $fd_s(k,p) = 0$. The cumulative minimum transmission time $t(s,p)$ and cumulative delay deviation $d(s,p)$ for all frames of a stream at a port are given by (2):

$$t(s,p) = \sum_{k=1}^{m} ft_s(k,p) \qquad d(s,p) = \sum_{k=1}^{m} fd_s(k,p) \tag{2}$$

The robustness parameter $\Gamma \in [0,1]$ controls how much of the delay deviation is protected in the schedule: $\Gamma = 0$ is optimistic, $\Gamma = 1$ is fully robust. The forwarding is sequential: for each stream, the transmission of frame i at the next port ($next\_port_{s,u,p} = p'$) starts only after frame i has been fully received and processed at the previous port (p), and after frame $i-1$ has finished transmission at p'. This ensures frames are forwarded in order and never overlap. The offset time required $next(\Gamma, s, u, p, p')$ between consecutive ports is calculated (5):

$$A_i(s,p) = \sum_{k=1}^{i} ft_s(k,p) + \min\{\Gamma \cdot d(s,p), \sum_{k=1}^{i} fd_s(k,p)\} \tag{3}$$

$$B(s,p) = \max_{i=2}^{m} \left\{A_i(s,p) - A_{i-1}(s,p')\right\} \tag{4}$$

$$next(\Gamma, s, u, p, p') = \max\{A_1(s,p), B(s,p)\} \tag{5}$$

Frames are buffered at wireless bridges until their scheduled transmission window, creating deterministic departure times for subsequent scheduling. The robustness parameter Γ directly scales the uncertainty budget $d(s,p)$, determining the required scheduling offset $next(\Gamma, s, u, p, p')$ between ports. Increasing Γ allocates larger time windows for wireless transmissions, improving reliability but reducing network capacity for other streams. This tunable approach balances protection against wireless delay variation and overall schedulability.

C. Integer Linear Program (ILP)

With decision variables (stream: s, path: u, port: p, repetition instance within LCM_S: r), $x^{r,u}_{s,p}$ is the transmission window starting times at ports for streams, ($z_{s,u}$, a_s) are for (path,

stream) selection (activation) and $y_{s,r,u,p}^{s',r',u'}$ is a variable to help building temporal isolation constraint (8):

$$x_{s,p}^{r,u} \in \{0, \mathbb{Z}^+\}, \quad \{y_{s,r,u,p}^{s',r',u'}, z_{s,u}, a_s\} \in \{0,1\}$$

The objective (left of (6)) is to maximize the number of scheduled streams, subject to the path selection constraint (right of (6)), which ensures that exactly one path is activated for each scheduled stream. Constraints, such as temporal isolation, latency, and jitter bounds, are detailed in constraints (7)–(11).

$$\max \sum_{s \in S} a_s \qquad \text{s.t.} \quad \sum_{u \in U_s} z_{s,u} = a_s \quad \forall s \in S \tag{6}$$

Constraint (7) enforces, given port p and next port is p' of a path u of stream s, p' time window is scheduled exactly after $next(\Gamma, s, u, p, p')$ (5) amount of time:

$$\begin{aligned} & x_{s,p}^{r,u} + next(\Gamma, s, u, p, p') \cdot z_{s,u} = x_{s,p'}^{r,u} \\ & s \in S, u \in U_s, r \in R_s, p \in P_{s,u}, \\ & p' = next\_port_{s,u,p}, p \neq \textit{last}(P_{s,u}) \end{aligned} \tag{7}$$

Constraint (8) isolates streams temporally at ports, considering robustness Γ and wireless delay deviations $d(s,p)$.

$$\begin{aligned} & \text{(a)} \quad x_{s,p}^{r,u} + (t(s,p) + d(s,p) \cdot \Gamma) \cdot z_{s,u} \\ & - x_{s',p}^{r',u'} \leq M \cdot y_{s,r,u,p}^{s',r',u'} \\ & \text{(b)} \quad x_{s',p}^{r',u'} + (t(s',p) + d(s',p) \cdot \Gamma) \cdot z_{s',u'} \\ & - x_{s,p}^{r,u} \leq M \cdot (1 - y_{s,r,u,p}^{s',r',u'}) \\ & s \in S, u \in U_s, r \in R_s, p \in P_{s,u}, \\ & (s',u') \in P'_p, r' \in R_{s'}, s < s' \end{aligned} \tag{8}$$

Constraint (9), if a path of a stream is scheduled ($z_{s,u}$ = 1), a time window allocated for each stream instance at the talker port $p = \textit{first}(P_{s,u})$ every C_s, ensuring cyclic instance creation.

$$\begin{aligned} & x_{s,p}^{r,u} \geq (r-1) \cdot C_s \cdot z_{s,u} \\ & s \in S, u \in U_s, r \in R_s, p = \textit{first}(P_{s,u}) \end{aligned} \tag{9}$$

Constraint (10), end-to-end latency of stream from time=0 of cycle period C_s is within the allotted $max\_latency_s$.

$$\begin{aligned} & x_{s,p}^{r,u} + (t(s,p) + d(s,p) \cdot \Gamma) \cdot z_{s,u} \leq \\ & max\_latency_s + (r-1) \cdot C_s \\ & s \in S, u \in U_s, r \in R_s, p = \textit{last}(P_{s,u}) \end{aligned} \tag{10}$$

Constraint (11), the jitter of a stream occurring from different instances r of the same stream is bounded by max_jitter_s.

$$\begin{aligned} & \text{(a)} \quad (x_{s,p}^{r,u} - (r-1) \cdot C_s) - (x_{s,p}^{r',u} - (r'-1) \cdot C_s) \\ & + M \cdot (1 - z_{s,u}) \geq -max\_jitter_s \\ & \text{(b)} \quad (x_{s,p}^{r,u} - (r-1) \cdot C_s) - (x_{s,p}^{r',u} - (r'-1) \cdot C_s) \\ & - M \cdot (1 - z_{s,u}) \leq max\_jitter_s \\ & s \in S, u \in U_s, p = \textit{first}(P_{s,u}), r \in R_s, r' \in R_s, r < r' \end{aligned} \tag{11}$$

D. How to obtain link latency bounds in practice?

Accurate wireless link delay bounds are essential but challenging, as radio frequency conditions and interference vary dynamically. Bounds can be obtained via empirical measurements (e.g., synchronized timestamps [16], [17]), vendor service level agreements (e.g., private 5G), or analytical methods like network calculus, though the latter may be overly conservative. Simulation-based profiling is useful in the design phase. In practice, combining offline profiling and online monitoring enables adaptive bounds, which is crucial in dynamic environments.

E. Sequential Batch Scheduling Heuristic

To address scalability, we propose a sequential batch scheduling ILP heuristic. Streams are partitioned into disjoint batches, and the ILP is solved iteratively for each batch. After each batch, scheduled streams are fixed and excluded from further optimization, reducing problem size but limiting global optimality. Batch size controls the trade-off between solution quality and computational time: smaller batches are faster but more myopic, while larger batches approach the exact ILP but are slower. Streams are sorted by scheduling priority, which reflects network preferences for which streams should be scheduled first. Thus, higher-priority streams are scheduled when resources are most available. Within each batch, the ILP schedules all streams without further prioritization.

In each iteration, fixed streams (from previous batches) have immutable paths and time windows, creating unavailable time blocks on ports. Variable streams (current batch) must avoid these blocks. To reduce constraints, consecutive fixed blocks too small for a variable stream are merged (see Figure 2).

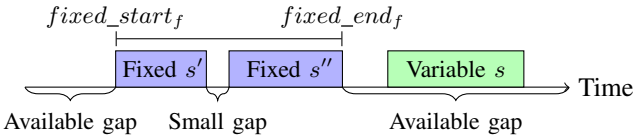

Fig. 2: $fixed\_start_f$ and $fixed\_end_f$ times for a variable stream

Let $fixed\_set(s,p)$ be a function that returns the set of merged fixed time blocks $(fixed\_start_f, fixed\_end_f)$ for a variable stream s at port p, representing unavailable temporal windows due to previously scheduled streams (see Figure 2). Constraint (12) ensures that the transmission window of a new variable stream does not overlap with any fixed block on the port; it must be scheduled entirely before or after each fixed block. Algorithm 1 outlines the sequential scheduling process.

$$\begin{aligned} & \text{(a)} \quad x_{s,p}^{r,u} + (t(s,p) + d(s,p) \cdot \Gamma) \cdot z_{s,u} \leq \\ & fixed\_start_f + M \cdot y_{s,r,u,p}^{f} \\ & \text{(b)} \quad fixed\_end_f \leq x_{s,p}^{r,u} + M \cdot (1 - y_{s,r,u,p}^{f}) \\ & s \in S, u \in U_s, r \in R_s, p \in P_{s,u} \\ & (fixed\_start_f, fixed\_end_f) \in fixed\_set(s,p) \end{aligned} \tag{12}$$

Algorithm 1: Sequential Batch Scheduling Heuristic

Input: Set of streams $\mathcal{S}$, number of batches B
Output: Fixed variables, $z_{s,u}$, a_s, $x_{s,p}^{r,u}$

```
Sort streams in S by priority into S_sorted;
Partition S_sorted into B disjoint batches: B_1, ..., B_B;
for i ← 1 to B do
    Set streams in B_i as variables, all previously
      scheduled streams as fixed;
    Reset variables y_{s,r,u,p}^{s',r',u'}, y_{s,r,u,p}^{f};
    Reset and re-add constraints (8) and (12) for B_i ;
    Solve the ILP for the current batch;
    Fix variables a_s, z_{s,u}, x_{s,p}^{r,u} for scheduled streams;
```

The computational complexity grows with the number of streams sharing a port, especially for variable streams. Fixed streams add smaller complexity, and merging fixed blocks reduces constraints as scheduling progresses. The heuristic scales polynomially by iterating over fixed streams to build sequential ILP models, with batch size limiting the complexity of ILPs. This enables efficient scheduling for large TSNs.

V. Evaluation

Our evaluation aims to answer the following questions:

1) How does the heuristic compare against the optimal? What is the impact of different link characteristics, network topologies, streams, and batch sizes? (cf. V-B)
2) What is the trade-off between robustness, wireless link latency deviations, and the number of scheduled streams? (cf. V-C)

A. Experimental Setup

1) Evaluation Platform: Experiments ran on a server with an Intel Xeon Gold 6326 CPU (32 cores) and 250 GB RAM. We used Gurobi Optimizer 12.0.1 with a 2-hour time limit for medium and small, and 4-hour for large per experiment to solve ILPs. For the exact ILP, this limit applies to the single run; for the batch heuristic, it is distributed equally across all batches. If the time limit is reached, the best feasible solution is used. The code and experiments we share for reproducibility.[1]

2) Network and Link Characteristics: Wired links use 100 Mbps, with $propagation\_delay$ and $processing\_delay$ set to 10μs (see (1)). For wireless, we use two datasets: URLLC [17] (latencies for 32B and 1420B packets, extrapolated for others; see Figure 3) and Det6G [16] (5G, long-tail delay, see Figure 4). Histograms determine $ft_s(k,p)$ (minimum observed latency) and $fd_s(k,p)$ (delay deviation, i.e., the difference between the upper and lower bounds of the link latency).

3) Traffic Scenarios: TSN streams are generated by randomly selecting properties from Table II. Each stream has $max\_latency_s$ and max_jitter_s that are proportional random multiples of C_s. Streams have priority levels 1–3 (equal distribution); priority 3 is highest. Det6G scenarios designed with longer C_s due to higher latencies.

[1]https://mygit.th-deg.de/inets/rctsn

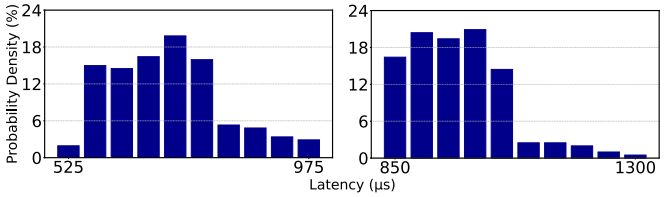

Fig. 3: Wireless link delay histograms for different packet sizes (URLLC scenario, left: 32B, right: 1420B) [17]

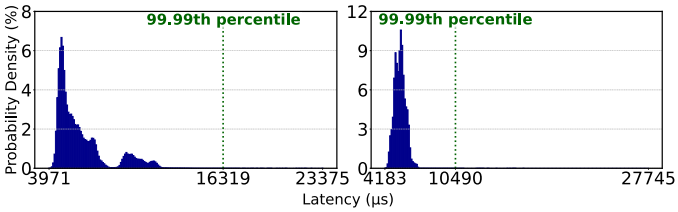

Fig. 4: Wireless link delay histograms for 100B packets (Det6G data, left: downlink, right: uplink) [16]

TABLE II: Traffic and Stream Parameters

C_s (URLLC)	$\{500, 1000, 2000, 4000, 8000\}$ μs
$packet\_size$ (URLLC)	$\{32, 64, 128, 256, 512, 1024, 1420\}$ Bytes
C_s (Det6G)	$\{20000, 40000\}$ μs
$packet\_size$ (Det6G)	$\{100\}$ Bytes
$max\_latency_s$	$\{0.5, 0.6, 0.7, 0.8, 1.0\}$ times C_s
max_jitter_s	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$ times C_s
$priority$	{1, 2, 3}, highest is 3, distributed equally

4) Experimental Design: We test three topologies: small (5 wired, 5 wireless bridges, ring), medium (50, 50), and large (220, 220), with random sub-topologies. For computational experiments, we use URLLC, varying batch size and stream count, fixing Γ=1. For the robustness evaluation, we used the small topology with both the URLLC and Det6G scenarios, employing only the exact ILP to ensure a fair comparison.

For each stream, up to $k = 3$ shortest paths are precalculated using Yen's algorithm [15]. Experimental stream sets are generated using the heuristic with small batches, then shuffled (except for priority, which is uniform). This biases the experimental sets toward less stringent requirements, but some tight-deadline streams remain. In theory, the exact model can schedule all streams given unlimited resources.

5) Key Performance Indicators: Streams scheduled (%) is the ratio of scheduled to requested streams; priority 3 scheduled (%) is the ratio for priority 3 streams. Port utilization (%) is the average fraction of time ports are scheduled to transmit in LCM_S. Utilization rises with more streams and longer paths. Success probability (%) is the cumulative delay distribution up to the chosen robustness level Γ and upper bound $d(s,p)$, representing the likelihood that wireless transmissions finish within the allocated window; for multiple wireless links, probabilities are multiplied.

B. Computational Experiments

We evaluate the heuristic and the exact ILP across small, medium, and large topologies (see Figure 5–7). The exact ILP is used as a baseline since it provides the optimal solution

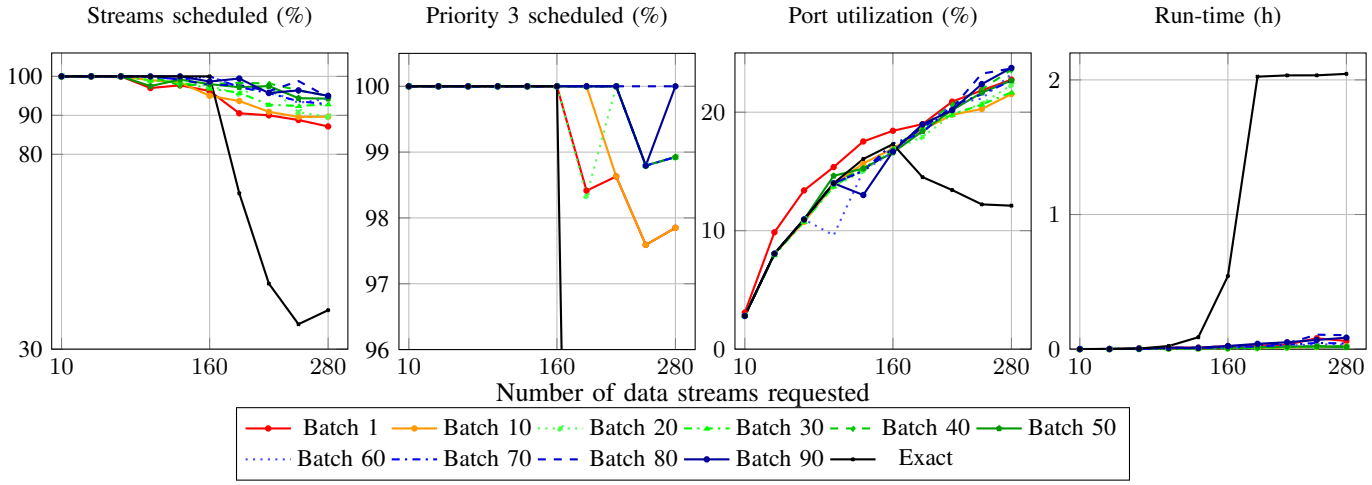

Fig. 5: Small topology computational experiments

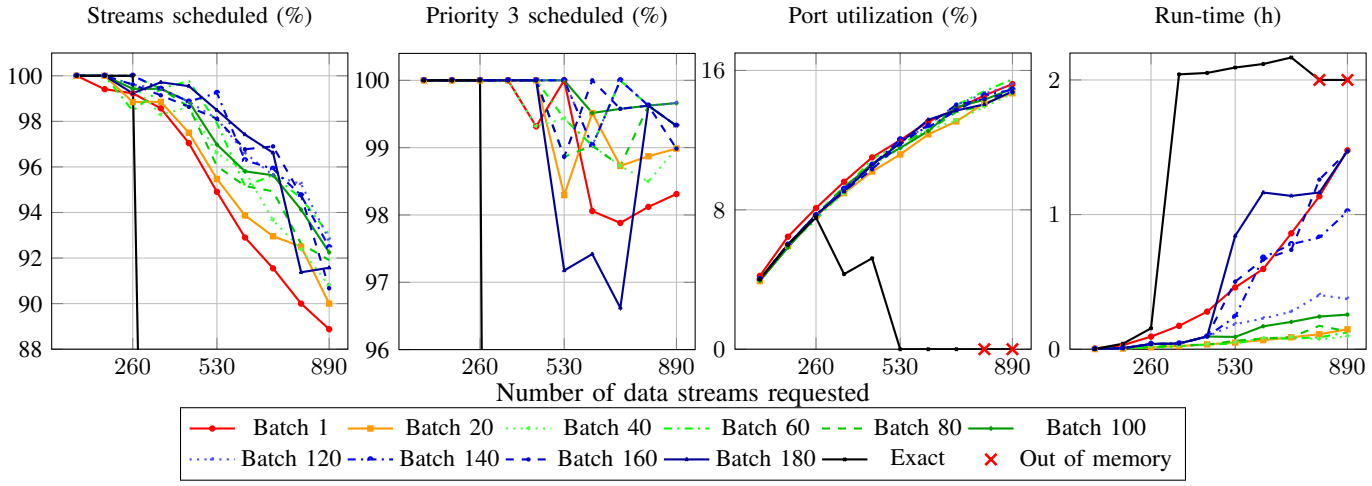

Fig. 6: Medium topology computational experiments

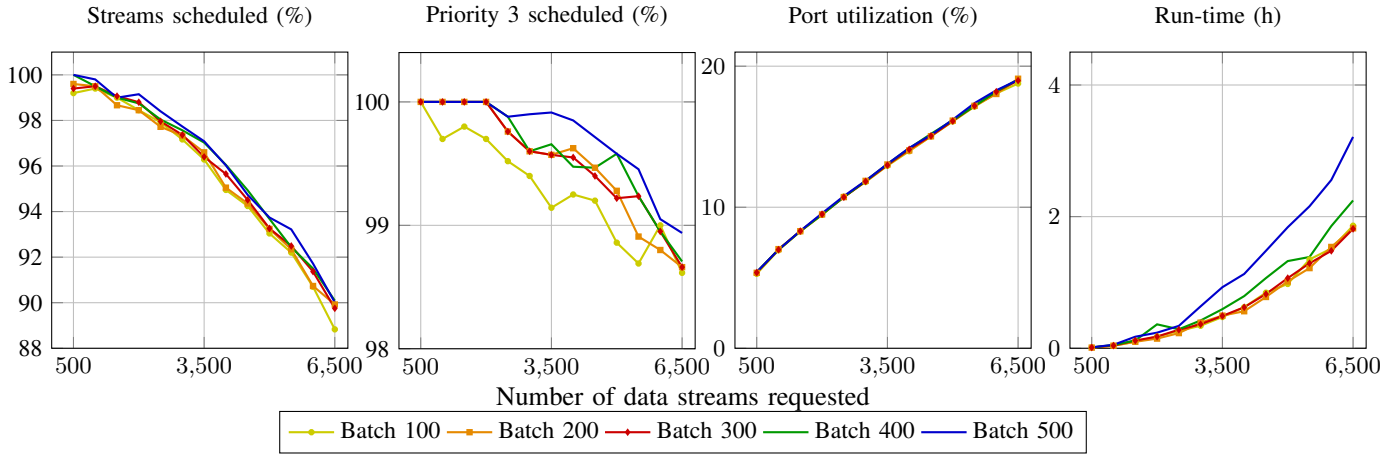

Fig. 7: Large topology computational experiments

when tractable, allowing us to directly assess the quality and scalability of the heuristic. However, the ILP quickly becomes intractable as problem size grows, making it suitable only for small and moderate scenarios.

In the small topology experiments (Figure 5), the exact ILP schedules all streams at low loads but struggles to find an optimal solution beyond 160 streams due to the time limit. The batch heuristic maintains efficiency as load increases.

For the medium topology (Figure 6), the ILP can schedule up to 260 streams before running out of memory or time, while the heuristic continues to scale, scheduling more streams as load increases. Larger batch sizes improve overall schedulability, but require more resources. Smaller batches yield higher port utilization by favoring longer, less-contended paths, while larger batches maximize total schedulability with shorter paths.

In the large topology (Figure 7), we schedule up to 6500 streams. Only the heuristic is evaluated due to the intractability of the exact ILP. The heuristic consistently schedules over 88% of requested streams, with nearly all high-priority streams admitted, and maintains a reasonable runtime of 2 hours for offline scheduling as the network scales.

Overall, the sequential batch ILP heuristic is predictable and scalable, reliably scheduling high-priority traffic, while the exact ILP is only practical for small instances.

C. Robustness Experiments

To evaluate robustness against wireless link latency deviations, we use two real-world 5G-like datasets: URLLC (low, deterministic latencies) and Det6G (long-tail delays). For each, we test 173 (URLLC) and 139 (Det6G) streams, with each stream traversing one or two wireless links. For Det6G, we consider upper bounds (UB) at 100th, 99.99th, and 99.9th percentiles. The robustness parameter Γ is varied from 0 to 1, and scheduling is performed using the exact ILP.

Figure 8 shows the percentage of streams scheduled (left) and the success probability (right) as Γ increases. As robustness increases, fewer streams can be scheduled due to larger, more conservative time windows for wireless transmissions, which reduces network capacity. However, the success probability—i.e., the likelihood that all deadlines are met despite wireless delay variations—increases accordingly. This illustrates the fundamental trade-off: higher Γ prioritizes reliability, while lower Γ maximizes network utilization. For Det6G, only the 100th percentile UB with Γ=1 achieves true 100% reliability, reflecting the impact of long-tail delay distributions.

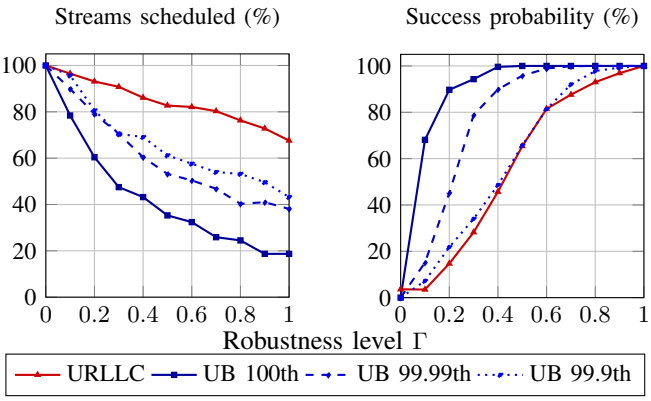

Fig. 8: Robustness: URLLC (173 streams), Det6G (139 streams)

VI. Conclusion and Future Work

This paper proposes a robust scheduling approach for TSN networks with wireless links, addressing latency uncertainties through a linear programming model and a scalable batch-scheduling ILP heuristic. The heuristic performs well for a moderate network load, with tunable batch size and robustness parameters that balance schedulability and reliability. In future work, we plan to explore clustering-based batch strategies, inclusion of best-effort traffic, cyclic stream dependencies, and stream-specific robustness based on traffic criticality.

References

[1] Time-Sensitive Networking (TSN) Task Group. [Online]. Available: https://1.ieee802.org/tsn/

[2] K. Zanbouri, M. Noor-A-Rahim, J. John, C. J. Sreenan, H. V. Poor, and D. Pesch, "A Comprehensive Survey of Wireless Time-Sensitive Networking (TSN): Architecture, Technologies, Applications, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 4, pp. 2129–2155, 2025.

[3] 3GPP, "Study on scenarios and requirements for next generation access technologies (Release 18)," Tech. Rep. 38.913, 2024. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2996

[4] Ö. O. Kaynak, A. Kassler, A. Fischer, O. Dobrijevic, and H. Chahed, "TSN Scheduling Robust to Wireless Performance Uncertainties: A Problem and Model Definition," in *Proc. of the KuVS Fachgespräch - Würzburg Workshop on 6G Networks (WueWoWAS'24)*, 2024, p. 4.

[5] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proc. of the 24th Int'l Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[6] F. Dürr and N. G. Nayak, "No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)," in *Proc. of the 24th Int'l Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[7] S. Egger, J. Gross, J. Sachs, G. P. Sharma, C. Becker, and F. Dürr, "End-to-End Reliability in Wireless IEEE 802.1Qbv Time-Sensitive Networks," in *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*, 2025, pp. 1–10.

[8] D. Ginthör, R. Guillaume, J. von Hoyningen-Huene, M. Schüngel, and H. D. Schotten, "End-to-end Optimized Joint Scheduling of Converged Wireless and Wired Time-Sensitive Networks," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 222–229.

[9] D. Ginthör, R. Guillaume, M. Schüngel, and H. D. Schotten, "Robust End-to-End Schedules for Wireless Time-Sensitive Networks under Correlated Large-scale Fading," in *2021 17th IEEE International Conference on Factory Communication Systems (WFCS)*, 2021, pp. 115–122.

[10] G. P. Sharma, W. Tavernier, D. Colle, M. Pickavet, J. Haxhibeqiri, J. Hoebeke, and I. Moerman, "End-to-End No-wait Scheduling for Time-Triggered Streams in Mixed Wired-Wireless Networks," *Journal of Network and Systems Management*, vol. 32, no. 3, p. 65, 2024.

[11] Z. Li, J. Yang, C. Guo, J. Xiao, T. Tao, and C. Li, "A Joint Scheduling Scheme for WiFi Access TSN," *Sensors*, vol. 24, no. 8, 2024.

[12] D. Bertsimas and M. Sim, "The Price of Robustness," *Operations Research*, vol. 52, no. 1, pp. 35–53, 2004.

[13] 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.

[14] R. Dinand, G. Eriksson, K. Wang, M. Matti, and J. Jeong, "Communication System with De-jitter Buffer for Reducing Jitter," US patent US 11,765,094 B2, Sep. 19, 2023.

[15] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[16] G. P. Sharma, D. Patel, J. Sachs, M. De Andrade, J. Farkas, J. Harmatos, B. Varga, H.-P. Bernhard, R. Muzaffar, M. Ahmed, F. Dürr, D. Bruckner, E. M. De Oca, D. Houatra, H. Zhang, and J. Gross, "Toward Deterministic Communications in 6G Networks: State of the Art, Open Challenges and the Way Forward," *IEEE Access*, vol. 11, pp. 106 898–106 923, 2023.

[17] P. Kehl, J. Ansari, M. H. Jafari, P. Becker, J. Sachs, N. König, A. Göppert, and R. H. Schmitt, "Prototype of 5G Integrated with TSN for Edge-Controlled Mobile Robotics," *Electronics*, vol. 11, no. 11, 2022.