network

Review

TSN Network Scheduling—Challenges and Approaches

Hamza Chahed [1,*] and Andreas Kassler [1,2]

[1] Department of Computer Science, Karlstad University, 656 34 Karlstad, Sweden; andreas.kassler@kau.se
[2] Intelligent Networks and Systems, Deggendorf Institute of Technology, 94469 Deggendorf, Germany
* Correspondence: hamza.chahed@kau.se

Abstract: Time-Sensitive Networking (TSN) is a set of Ethernet standards aimed to improve determinism in packet delivery for converged networks. The main goal is to provide mechanisms that enable low and predictable transmission latency and high availability for demanding applications such as real-time audio/video streaming, automotive, and industrial control. To provide the required guarantees, TSN integrates different traffic shaping mechanisms including 802.1Qbv, 802.1Qch, and 802.1Qcr, allowing for the coexistence of different traffic classes with different priorities on the same network. Achieving the required quality of service (QoS) level needs proper selection and configuration of shaping mechanisms, which is difficult due to the diversity in the requirements of the coexisting streams under the presence of potential end-system-induced jitter. This paper discusses the suitability of the TSN traffic shaping mechanisms for the different traffic types, analyzes the TSN network configuration problem, i.e., finds the optimal path and shaper configurations for all TSN elements in the network to provide the required QoS, discusses the goals, constraints, and challenges of time-aware scheduling, and elaborates on the evaluation criteria of both the network-wide schedules and the scheduling algorithms that derive the configurations to present a common ground for comparison between the different approaches. Finally, we analyze the evolution of the scheduling task, identify shortcomings, and suggest future research directions.

Keywords: Time-Sensitive Networking (TSN); scheduling; shaping; network configuration; Time-Aware Shaper (TAS); Cyclic Queuing and Forwarding (CQF); asynchronous traffic shaping (ATS)

check for updates

Citation: Chahed, H.; Kassler, A. TSN Network Scheduling—Challenges and Approaches. *Network* **2023**, *3*, 585–624. https://doi.org/10.3390/network3040026

Academic Editor: Martin Reisslein

Received: 14 November 2023
Revised: 8 December 2023
Accepted: 11 December 2023
Published: 16 December 2023

1. Introduction

Industrial IoT applications typically have stringent requirements on the dependability and performance of communication networks. For such applications, it is crucial that the network can provide latency guarantees, even under link and switch failures. Therefore, industrial networks must provide communication paths that have deterministic properties of controlled latency, low packet loss, low packet jitter, and high reliability. However, legacy Ethernet-based networks can only provide a best-effort delivery service, which led to the emergence of proprietary network protocols in the OT (operation technology) layer such as Profinet [1], preventing the open flow of the information to the IT (information technology) layers. To cope with the lack of determinism in traditional Ethernet and the obscurity of proprietary protocols, a set of amendments to the IEEE 802.1Q standard Ethernet has been specified within the Time-Sensitive Networking (TSN) community that aims to support the delivery of real-time traffic having stringent time constraints over converged networks.

Several surveys have been conducted in recent years to explore various aspects of TSN standards, including basic concepts, core functions, forwarding management, applications, and technologies [2–8]. However, we emphasize the need for more attention to be given to the shaping mechanisms and their configurations. These mechanisms play a crucial role in determining TSN network performance in terms of latency, jitter, and reliability. Traffic shaping in the TSN context involves prioritizing certain frames and introducing a queuing delay to establish a desired traffic profile [9]. Each frame experiences latency while waiting in the queue for its designated transmission time slot. Therefore, achieving guarantees on latency and jitter requires careful configuration of the transmission process. Selecting the appropriate shaping option requires a clear understanding of traffic requirements and characteristics, as well as the use of suitable algorithms to derive a coherent network-wide configuration. The literature proposes various algorithms based on different assumptions and optimization objectives and satisfying different constraints. However, the lack of common evaluation criteria for network schedules and scheduling algorithms adds to the complexity of making the right choice. Therefore, we believe that a comprehensive analysis of the different choices related to the scheduling problem, including optimization objectives, constraints, and assumptions, would be valuable to the community. This analysis would provide a solid foundation for making informed decisions regarding the selection and configuration of shaping mechanisms using suitable scheduling algorithms.

The primary contributions of this survey include addressing the limitations of various shaping mechanisms and establishing a connection between traffic types and potential shaping mechanisms. Specifically focusing on Time-Aware Streams and their associated shaping mechanism, Time-Aware Shaping (TAS), we propose a methodology that enables a fair comparison of different network scheduling approaches. We begin by presenting and analyzing recent research on network-wide scheduling, highlighting advancements, current challenges, and future research directions. We delve into the optimization goals discussed in the literature and examine their impact on schedule quality. Furthermore, we explore the scheduling constraints and assumptions presented in previous works. Emphasizing determinism, we investigate the underlying causes of nondeterminism. Building on this analysis, we develop a methodology for benchmarking schedule quality by defining a set of evaluation criteria that facilitate comparisons among different approaches.

The paper is organized as follows. Section 2 summarizes the background and provides references to surveys presenting the larger TSN context. In Section 3, we discuss the different shapers, their configuration, and their limitations. In Section 4, we review the state of the art concerning the scheduling approaches, analyze the challenges, and elaborate on future research directions. We deeper examine the TAS-scheduling in Section 5 concerning the objectives, constraints, and assumptions. We elaborate on network schedule quality benchmarking in Section 6 and conclude our paper in Section 7.

2. Background and Related Work

In this section, we begin by conducting a comprehensive review of relevant TSN-related surveys to provide a broader understanding of the context for this work. We then proceed to present an overview of various traffic categories, highlighting their distinctive characteristics. Additionally, we discuss the diverse traffic isolation techniques that can be employed in converged TSN networks. By addressing these aspects, we aim to establish a solid foundation for the subsequent discussions and analyses in this study.

2.1. Overview on TSN Related Surveys

TSN, as a large and continuously evolving set of standards, poses challenges for adopters seeking to understand the state of the art, identify missing features, and effectively utilize the standards for their specific use cases. Consequently, several surveys have been conducted to provide a comprehensive overview of TSN, which we summarize in Table 1. These surveys encompass a range of approaches, including general TSN overviews [2,4,5] and more specialized surveys focusing on specific aspects [3,6–8].

Among the general TSN overview surveys, Nasrallah et al. [4] stands out for categorizing TSN standards into five topics, such as flow concept, synchronization, management, control, and integrity. The survey also explores ultra low-latency (ULL) applications, their latency and jitter requirements, and shaping techniques beyond the TSN standard e.g., Burst Limiting Shaper (BLS). A distinguishing feature of this survey is its inclusion of TSN alongside Detnet and 5G, providing insights into standardization efforts and research directions.

In contrast, Bello et al. [5] groups TSN standards into four clusters based on their targets: timing and synchronization, reliability, low bounded latency, and resource management. This categorization aids network designers in selecting the relevant parts based on their specific use cases. Additionally, the survey provides an overview of Ethernet concepts, including the IEEE 802.1Q standard, which encompasses both the TSN and pre-TSN aspects. The paper also references studies that explore TSN applications in various fields with a focus on industrial automation and automotive industries. Notably, this survey discusses the merits and challenges of TSN, scoring nine requirementsand highlighting areas where TSN excels, such as bandwidth and scalability, as well as areas where it has limitations, such as confidentiality, integrity, and ease of configuration. The authors also identify configuration synthesis as a demanding research direction.

In response to the identified need for more detailed understanding and insights, our survey aims to complement the existing TSN-related landscape by addressing the challenges pertaining to schedule synthesis and providing an in-depth exploration of related work.

Contrary to our work, Seol et al. [2] categorizes works into three main topics: TSN core functions, network management, and resources for researchers. The survey provides a summary of studies related to the internal functioning of TSN, explores the interaction between entities within a TSN domain, and offers information on existing projects, tutorials, hardware, and software frameworks relevant to TSN research. It also includes a statistical analysis to identify research trends and areas that require further investigation.

Table 1. An overview of the presented TSN-related surveys.

Topics	[2]	[4]	[5]	[8]	[7]	[6]	[3]
Basic concepts		✓		✓			
TSN core functions	✓	✓	✓				
Forwarding management	✓	✓					
Application fields		✓	✓		✓		✓
Coordinating technologies		✓					
Tools and projects	✓						
Shaping and scheduling						✓	

There are other notable works that focus on specific aspects of the TSN landscape. For instance, in [8], the basics of traffic planning in time-triggered networking are explained, covering concepts like time-triggered communication, constraints, and the relationship between problem complexity, the number of frames/nodes, and constraints. The paper also discusses the evolution of planning tasks from classical time-triggered communication to TSN. Additionally, it explores the trade-off between schedule calculation time and the number of windows to schedule when using a general-purpose solver like SMT.

In [7], TAS is compared to Burst Limiting Shaper (BLS) and Peristaltic Shaper (PS), a slightly different version of Cyclic Queuing and Forwarding (CQF). However, this study lacks coverage of important works published after 2015. A more recent study, [6], compares two shapers, TAS and Adaptive Time-Aware Shaper (ATS). It provides a general overview of TSN with detailed information on ATS and TAS parameters, state variables, and main algorithms. The study proposes and evaluates two algorithms, adaptive bandwidth sharing and adaptive slotted window, to make TAS adaptive to sporadic traffic. However, the adaptive version of TAS may increase control traffic overhead. While study [6] focuses on shaping and scheduling, our work has a broader scope, covering the general case of shaping and scheduling, explaining basic concepts, and surveying and classifying a wider range of works.

In [3], the integration of TSN in automotive embedded systems is explored. The authors provide an overview of TSN amendments and their relevance to automotive applications such as assisted driving and fully autonomous driving. They identify relevant standards for in-vehicle networks and discuss the challenges and opportunities of integrating TSN into the in-vehicle model-based development process.

The focus in our survey lies specifically on the scheduling problem within TSN, defined as finding the optimal path and shaper configurations for all TSN elements (e.g., in terms of transmission slots) to provide the required QoS to all scheduled streams [8]. We analyze various approaches and consider TAS scheduling as it has garnered significant attention and is particularly challenging. Moreover, we extend our coverage to network-wide scheduling issues and offer suggestions for future research.

While our survey does not attempt to cover the entirety of TSN research, Figure 1 provides an overview of our paper's structure and its place within the broader TSN landscape. We acknowledge the importance of addressing other crucial aspects related to scheduling, including control plane design and associated challenges, configuration workflow and overhead, and cross-domain configuration challenges. Although beyond the scope of this survey, these areas warrant attention and further exploration.

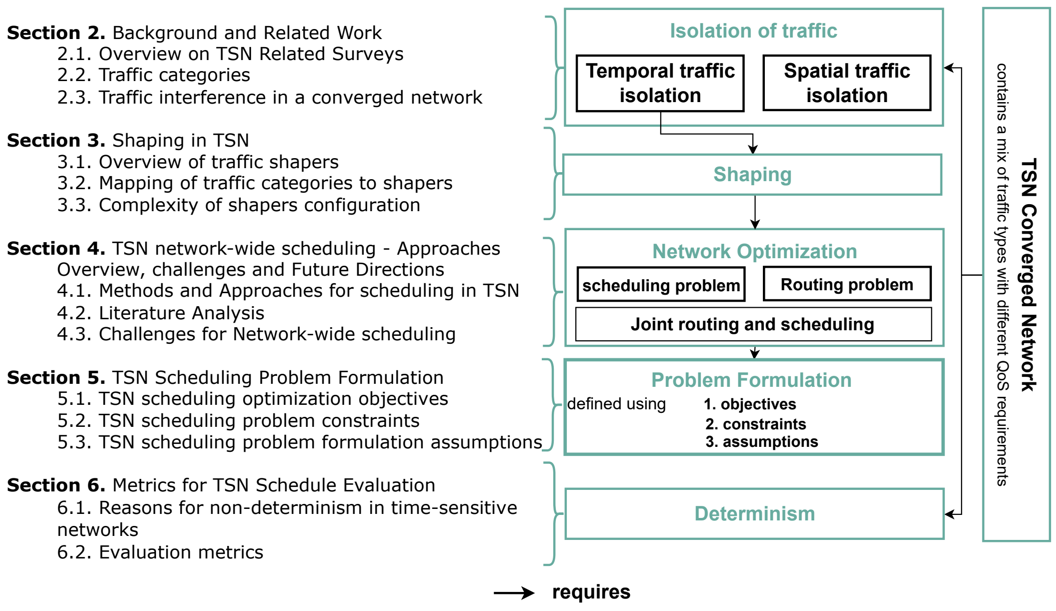

Section 2. Background and Related Work
2.1. Overview on TSN Related Surveys
2.2. Traffic categories
2.3. Traffic interference in a converged network
Section 3. Shaping in TSN
3.1. Overview of traffic shapers
3.2. Mapping of traffic categories to shapers
3.3. Complexity of shapers configuration
Section 4. TSN network-wide scheduling - Approaches Overview, challenges and Future Directions
4.1. Methods and Approaches for scheduling in TSN
4.2. Literature Analysis
4.3. Challenges for Network-wide scheduling
Section 5. TSN Scheduling Problem Formulation
5.1. TSN scheduling optimization objectives
5.2. TSN scheduling problem constraints
5.3. TSN scheduling problem formulation assumptions
Section 6. Metrics for TSN Schedule Evaluation
6.1. Reasons for non-determinism in time-sensitive networks
6.2. Evaluation metrics
Isolation of traffic
Temporal traffic isolation
Spatial traffic isolation
Shaping
Network Optimization
scheduling problem
Routing problem
Joint routing and scheduling
Problem Formulation
defined using
1. objectives
2. constraints
3. assumptions
Determinism
TSN Converged Network
contains a mix of traffic types with different QoS requirements
requires

Figure 1. An overview of the paper structure and how it fits into the TSN landscape.

2.2. Traffic Categories

Especially within the industrial context, there are different traffic types, which can be grouped into four categories based on three main traffic characteristics (see Table 2 [10]):

- Traffic regularity: defines if the traffic sending is periodic (cyclic) or irregular (acyclic);
- Timeliness: defines if Ethernet packets are sent at predefined or scheduled times (time-triggered) or not;
- Delivery mode: defines the requirements for receiving the packets in terms of maximum latency or if the packets should obey a given deadline.

For the convenience of the analysis throughout this paper, these categories can be organized according to their periodicity, and in that case, we refer to them as streams and nonstreams, or according to their time sensitivity, namely, HTS (high time-sensitive) for Time-Aware Streams (i.e., that are cyclic, time-triggered, and have tight timing constraints on their delivery), LTS (low time-sensitive) for the streams that require higher QoS than the BE traffic but not to the level of the HTS streams, and BE (Best-Effort). A similar grouping, by sensitivity, was introduced in [11], from which we adopted the naming.

Table 2. Summary of traffic categories.

Traffic Category	Traffic Characteristic			Traffic Type
	Regularity	Timeliness	Delivery	
Time-Aware Stream (HTS)	Cyclic	TT	Deadline	Isochronous
			Latency	Cyclic Synchronous
Stream (LTS)	Cyclic	ET	Latency	Cyclic Asynchronous
Traffic-Engineered nonstream (LTS)	Optional	ET	Latency	Network Control
	Acyclic			Alarms and Events
				Configuration and Diagnostics
Nonstream (BE)	Acyclic	ET	N/A	Best-Effort

TT = Time-triggered, ET = Event-triggered, N/A = not applicable. HTS = High time-sensitive, LTS = Low time-sensitive, BE = Best-Effort.

2.3. Traffic Interference in a Converged Network

Traffic isolation is needed to protect HTS traffic from interfering with other traffic to provide the required guarantees on timeliness. Isolation could be in the time domain (referred to as temporal isolation) using shaping (other techniques include frame-preemption [12] and guard-banding [13]) or spatially (referred to as spatial isolation) using routing and queuing in different queues. In general, both techniques can be combined for finding better solutions (e.g., solving a joint routing and scheduling optimization problem such as [14]) or by fixing some aspects of the overall problem to achieve a simpler problem formulation. Examples are to solve the scheduling problem for a given traffic path (routing problem is solved first), or to study the effect of routing HTS traffic over its scheduling (scheduling problem is solved first [15]). Finally, some works such as [16,17] propose a traffic isolation model based on network calculus where they protect the traffic from interference without the need for strict isolation techniques (i.e., usage of a different queue for every stream and coordinate the transmission time of the different flows). In the rest of the paper, we focus on shaping due to its importance as a temporal traffic isolation technique and the novelties brought to it by the TSN effort.

3. Shaping in TSN

3.1. Overview of Traffic Shapers

One shaping mechanism commonly used in traditional Ethernet is Strict Priority (SP) [18], which selects packets for transmission based on their queue priority. While successful for non-time-sensitive traffic, it lacks sufficient QoS guarantees for time-sensitive applications. Without proper admission control, real-time video streaming, for instance, can experience unpredictable delays due to network congestion. Network calculus techniques, as mentioned in [16], can estimate worst-case latency if flow data rates and burst sizes are known, aiding admission control decisions. However, latency variance can increase burst sizes, leading to the continuous accumulation of frames and potential QoS violations, known as the snowball effect.

To address this issue, the AVB task group (later renamed TSN task group) introduced the credit-based shaper (CBS) [19]. CBS assigns a credit counter to each CBS queue, allowing transmission only when the credit is positive. This mechanism reintroduces gaps between frames, reducing burst accumulation and robustly improving audio/video transmission quality. However, the shared nature of the credit pool among different streams does not fully solve the snowball effect. According to IEEE 802.1Qav [19], the credit is increasing based on the sum of the data rate of all streams. Therefore, frames of one stream can be sent back-to-back using the credit accumulated by the frames of another stream without waiting, undermining the gap introduction concept.

Asynchronous Traffic Shaping (ATS) [20] overcomes the snowball effect by allocating a token bucket to each stream. Upon frame arrival, the decision to transmit or to place on wait depends on the availability of tokens and ongoing transmissions. However, frames waiting longer than the maximum residence time are discarded. ATS also addresses the head-of-the-queue blocking problem by queuing frames based on ingress port and traffic class, resulting in smoother traffic patterns and reduced congestion loss. While ATS resolves the burst-size problem and offers per-hop latency bounds, it still falls short of meeting stringent latency and jitter requirements for Time-Aware Streams due to its asynchronous nature.

The TSN Task group introduced two more shaping strategies which are the Cyclic Queuing and Forwarding (CQF) [21] and the Time-Aware Shaper (TAS) [13]. Both of them are synchronous and require time synchronization through all the involved network elements. CQF defines a cycle time of fixed duration d that starts and ends at the same moment in all the network elements. Every frame that is received and queued by a switch during cycle i is transmitted to the next switch during cycle $i + 1$. Consequently, a frame traversing a path of n hops would experience a bounded latency of maximum $(n + 1) * d$ and minimum $(n - 1) * d$. These latency bounds hold as long as every switch in the path forwards the frame in the reserved cycle. In such a case, the actual queuing delay is not relevant.

The QoS offered by CQF is sufficient for some traffic types but it does introduce extra delay that can not be tolerated by HTS traffic besides having loose control on the jitter. TSN standard defines Time-Aware Shaping, as a more general and fine-grained mechanism but under the cost of more complex scheduling. TAS installs a gate on each of the eight egress queues of a port and controls the state of the gate to allow or forbid the transmission of frames. If the gate is closed, then no transmission from the associated queue is possible, and if the gate is open, then frames from the associated queue could be transmitted. In the case that there is another frame in transmission or the opening time of the gate is not enough to fully serialize the frame on the link, the transmission is blocked even if the gate is in the open state. The states of the gates are independent of each other and can be simultaneously opened or closed. They are controlled by a Gate Control List (GCL). Every entry in this list is a tuple, i.e., a timestamp and a state value (O—opened, C—closed). The GCL configuration requires a base time as well, which is a timestamp that defines when to start applying it. Once configured, the port runs through the GCL and applies the states at their respective timestamps until it reaches the end. Then it starts over from the beginning.

Unlike CQF, TAS can control the transmission timing of the frames to the queue level, which enables, if properly configured, the temporal isolation of the different traffic types. CQF is much simpler to configure compared to TAS and it tolerates variance in the arrival time of the frames. In addition, it does not require the traffic to be time-triggered as is the case with TAS, which allows CQF to handle cyclic and acyclic streams.

An important direction in future research is to study the benefits and pitfalls of the coexistence of different shaper types in the network. For example, combining TAS and CQF shaping mechanisms was investigated in [11]. However, a more detailed analysis is needed to fully understand the implications of different shaper coexistence in the network.

3.2. Mapping of Traffic Categories to Shapers

In this section, we discuss possible mappings between the presented traffic categories and the different shapers (e.g., SP, CBS, ATS, CQF, and TAS). Selecting a proper shaper and its configuration depends not only on the traffic mix of the use case but also on the traffic load and the available resources in the network (i.e., number of queues, queue length, bandwidth, buffer size, etc.).

Intuitively, shaping is about organizing the temporal and spatial distribution of the resources among the streams. Therefore, if the resources exceed by far the need of the streams there would be less interference, and consequently, a loose traffic isolation would be enough. For example, if we place a single HTS flow in a network that does not serve any traffic, all the resources will be available for the flow. If we want to add a BE stream, it might be enough to use SP, assign to the HTS stream the highest priority, and preempt the BE stream whenever a frame of the HTS stream is ready for transmission. Frame-preemption might be not needed if the transmission speed of the egress ports is so high (e.g., 10 Gbps) that the queuing delay of the HTS frame is not negatively impacting the deadlines, even if it has to wait for the transmission of a BE MTU. On the other hand, if the network is short of resources, it might not be possible to serve all the required streams. In this case, the choice of the shaping mechanism is not important since some of the streams would starve anyway.

We conclude from the above example that the first step in making the right choice is to have a clear understanding of the use -case and the available resources. However, a network designer would scale the network properly such that all the streams are served with the minimum investment. Therefore, in the rest of this subsection, we disregard the extreme cases of under-/overprovisioning and we consider only networks with reasonable resource utilization.

BE traffic does not require any QoS guarantees and any shaper is applicable (for example SP, which is the easiest to configure). For all the other traffic types, SP needs to be combined with at least another shaper, which needs to provide the required QoS by managing the queues properly. SP then works as a backup shaper in case multiple queues are eligible for transmission at the same time.

LTS traffic is event-triggered (i.e., non-time-triggered). Therefore, TAS is not suitable since it needs to configure the offset of the streams in the talker. For the cyclic LTS streams, CQF could be applied since it is by design operating cyclically, provides latency guarantees, and is convenient for streams with random offsets. For the acyclic nonstreams, CQF is not the most convenient as it needs to reserve slots for the streams. However, reserving transmission slots for acyclic streams wastes bandwidth that could be used to serve other traffic types. ATS or CBS, on the other hand, are more convenient and the best choice depends on the traffic load. As ATS, unlike CBS, attributes a different bucket for every stream, the queue management is on stream level. To provide the required guarantees on latency, appropriate queue configuration needs to be derived using, e.g., network calculus. However, in use cases involving time-critical events that have the highest priority (e.g., emergency traffic such as sending an event resulting from pressing a safety red button), ATS should be avoided as emergency traffic is sporadic and event-triggered. For such traffic, pre-reservation of resources is the safest option with existing shapers (e.g., study [22] proposed enhancements to TAS to support emergency traffic). However, enabling frame-preemption and assigning the highest priority queue exclusively to the emergency traffic (exclusive assignment is needed to avoid head-of-the-queue blocking) could be an interesting solution.

Finally, HTS has two traffic types (isochronous and cyclic synchronous), which can be distinguished by the delivery mode. For latency-based delivery mode, minimizing jitter is important, which should be as close to zero as possible. As HTS is time-triggered, TAS is a suitable shaper, while asynchronous shapers are not suitable because TAS can trigger the transmission when appropriate and avoids extra latency during transmission while asynchronous shapers do not have that option. As CQF is a special case of TAS, it might be applicable in certain use cases. Table 3 shows an overview of the suitability of different traffic shapers for the different traffic types.

3.3. Complexity of Shapers Configuration

Assuming network paths are fixed, the scheduling problem complexity involves the finding of a configuration for the given shapers to provide the desired QoS guarantees required by the traffic mix. This depends on the choice of the shaping mechanism. Configuring SP has the lowest complexity as it only requires assigning a priority to each queue. CBS and ATS, on the other hand, require the tuning of credit/tokens-related parameters which is still simple, but adequate for the audio/video domain as such flows typically do not have very strict latency requirements.

However, configuration complexity increases when using mechanisms aimed at synchronous traffic. For example, CQF requires the optimization of four main parameters:

- Cycle duration: A long cycle duration may increase jitter and violate the maximum latency requirement of the streams. On the other hand, a short cycle duration may be insufficient for transmitting the frame during the allotted slot.
- The alignment of the start and end times of the cycle among the involved switches: All switches must agree on the starting time of the cycles, which must be enforced by a centralized or decentralized configuration.
- The cycle boundaries within a switch: As the reception of a frame happens during a time interval, the receiving and the sending switches shall assign this time interval to the same given cycle. Otherwise, the forwarding of the receiving switch would be delayed by an additional cycle.
- The earliest and the latest times that a switch could transmit a frame within a cycle: This is important to minimize the impact of time synchronization errors. If TSN switches are not properly synchronized, a switch may send a frame at cycle i and the next switch may receive it during cycle $i-1$ or cycle $i+1$. This may lead to a situation where the accumulated end-to-end latency could violate the requirements.

Proper configuration for TAS requires the calculation of the Gate Control List (GCL) for each switch. This involves identifying the GCL cycle, the base time that identifies when to start applying the GCL, the time offsets for triggering the sending of the HTS streams, the state (open or closed) of the gates at every moment, and the states transition timestamps. TAS scheduling is proven to be an NP-hard optimization problem [23]. Therefore, we focus the rest of this survey on analyzing and discussing the different solution approaches from the literature.

Table 3. Traffic class to shaper mapping.

Traffic Category	**Shaping Option**				
	SP	**CBS**	**ATS**	**CQF**	**TAS**
HTS (isochronous)					✓
HTS (cyclic synchronous)				✓	✓
LTS streams			✓	✓	
LTS nonstreams		✓	✓	✓	
BE	✓				

4. TSN Network-Wide Scheduling—Approaches Overview, Challenges, and Future Directions

In this section, we aim to provide a comprehensive overview of the network scheduling task within the TSN field. We start by presenting the works that have been carried out from 2016 until November 2022. Then, we analyze those works to understand the evolution of scheduling during the last few years. Finally, we discuss the TSN scheduling challenges and possible future directions.

4.1. Methods and Approaches for Scheduling in TSN

The scheduling problem in TSN aims to find the proper paths and configuration of shapers to fulfill the QoS requirements of different flows. Finding configurations by solving the scheduling problem in TSN has been an important research topic during the last few years. We summarize in Appendix A the different contributions in Tables A1 and A2. We deployed the "quick and dirty" search method in Google Scholar to gather an initial set of papers, then we used the "Snowballing" search method using databases such as Scopus and Web of Science. For conciseness, we focus on works published in 2016 and afterward. The tables present only scheduling solutions and exclude works that are not touching the problem directly. Other works treat related topics such as the effect of HTS scheduling on other traffic types [24,25], comparing the performance of the different shapers [6,26,27], treating the consistency [28] or stability [29] issues of the network, optimizing the topology given the network schedule [30], optimizing the assignments of queues [31], the limitation of joint routing and scheduling, etc. To help the classification of the state of the art, Tables A1 and A2 include columns identifying the task (i.e., S for scheduling, R for routing, T for topology optimization, and A for task or application placement), the selected approach, whether it is considering static scheduling (i.e., off for offline) or scheduling at runtime (i.e., on for online), whether it uses a distributed scheduling (i.e., D in the tables) or centralized scheduling (i.e., C in the tables) algorithm, and what shaper the work focuses on. As can be seen from the tables, most of the works treat scheduling in the TAS context. However, some works combine it with CQF (e.g., [11]), treated CQF (e.g., [32,33]), or ATS (e.g., [34]) separately, or suggest enhancements to the shapers (e.g., study [35] proposed a TAS scheme with virtual queues (VQTAS), study [36] suggested a protection band to support emergency traffic, and study [6] suggested modifications to TAS to make it adaptive in a distributed approach).

The developed approaches can be classified broadly into two main categories, as depicted in Figure 2. The first category focuses on the exact resolution of the scheduling problem through mathematical formulation as an optimization problem such as SMT/OMT, CP, and (M)ILP formulation and uses off-the-shelf solvers. (e.g., [29,37–43]) to find a feasible or optimal solution. Solutions in the second category, including [34,44–49], develop heuristics or metaheuristics to solve the scheduling problem. While some of the works only tackle the scheduling problem (e.g., [34,37,38,41]), other approaches such as [40,43,46,50,51] treat the combined scheduling and routing problem to gain more flexibility in the solution space. Several works treat the scheduling problem as an offline problem (e.g., [52–56]) while other works aim to develop online algorithms for runtime scheduling (e.g., [44,57]). Nayak et al. [58] proposes both online and offline scheduling, while other works such as [59–61] calculate offline schedules, taking into consideration their effect on online scheduling. Some approaches aim for scalable solutions by fragmentation of the problem such as [48,62]. Others, such as [11,24], aim to improve LTS and BE QoS within TSN networks while serving HTS traffic. Finally, a few papers such as [34,63] propose distributed approaches, but most of the papers opt for calculating the schedules in a centralized fashion. Approaches such as [64] monitor the forwarding plane to repair broken schedules or to learn the traffic characteristics and enable the self-configuration of the network (e.g., [44]).

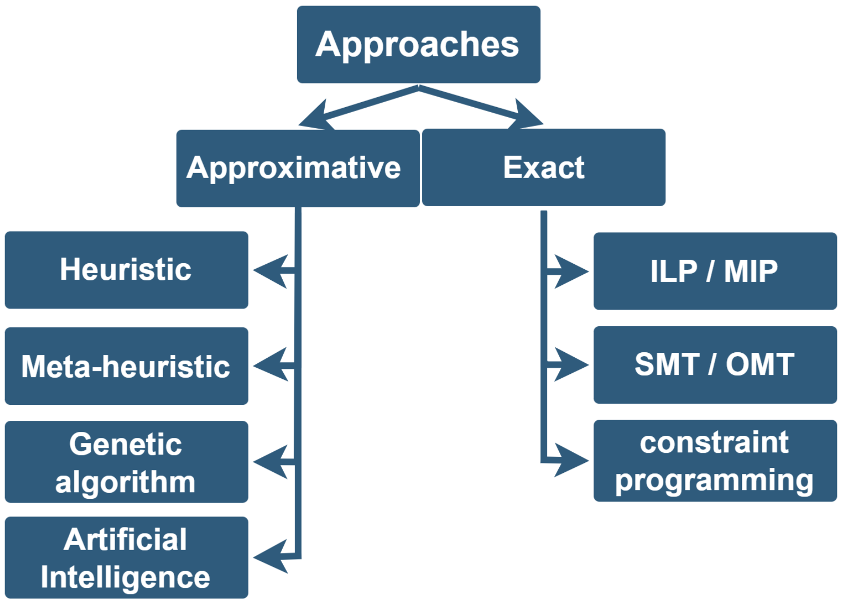
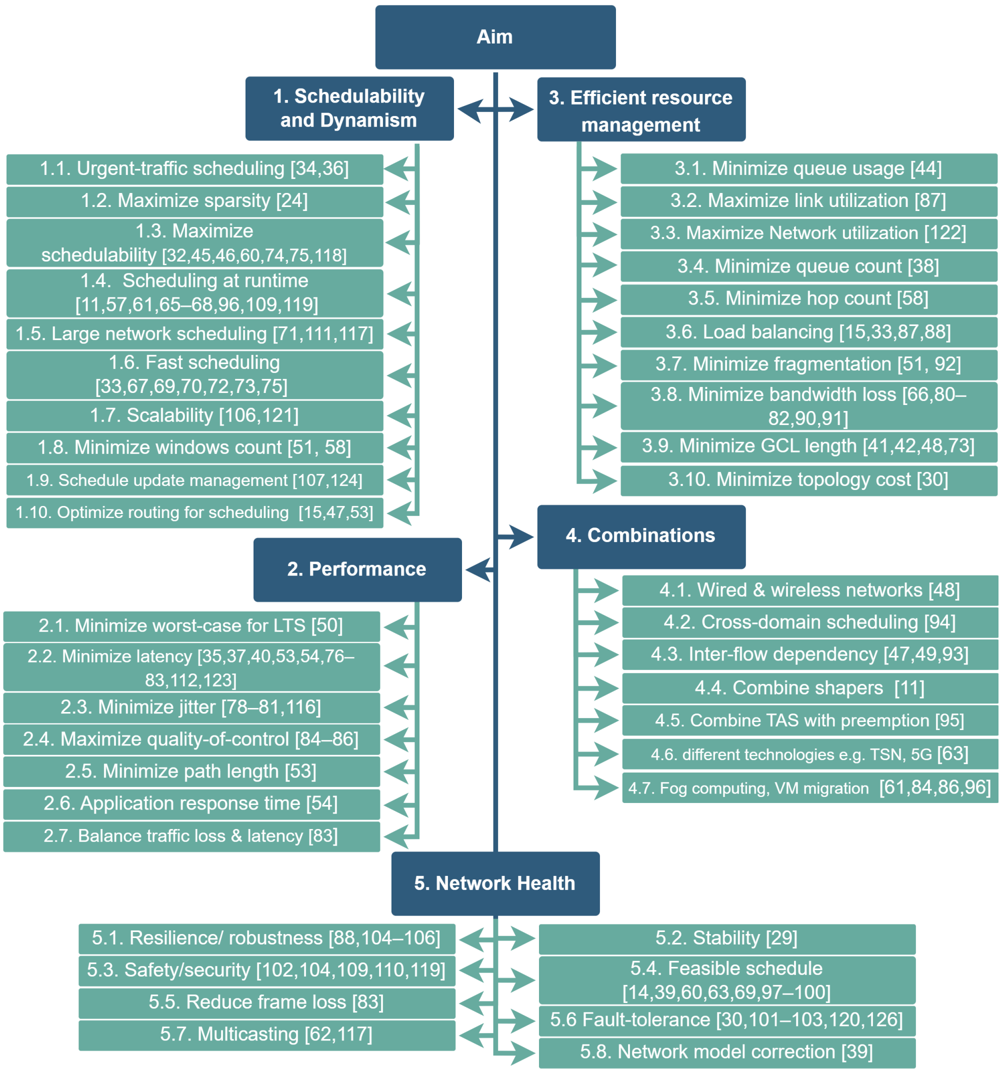


Figure 2. Classification of approaches.

For better clarity, Figure 3 summarizes and classifies the different optimization objectives of the different approaches. Note that some works may fall into multiple optimization objectives that could belong to the same category or different categories. In the following, we discuss each category separately.



Aim
1. Schedulability and Dynamism
1.1. Urgent-traffic scheduling [34,36]
1.2. Maximize sparsity [24]
1.3. Maximize schedulability [32,45,46,60,74,75,118]
1.4. Scheduling at runtime [11,57,61,65–68,96,109,119]
1.5. Large network scheduling [71,111,117]
1.6. Fast scheduling [33,67,69,70,72,73,75]
1.7. Scalability [106,121]
1.8. Minimize windows count [51, 58]
1.9. Schedule update management [107,124]
1.10. Optimize routing for scheduling [15,47,53]
3. Efficient resource management
3.1. Minimize queue usage [44]
3.2. Maximize link utilization [87]
3.3. Maximize Network utilization [122]
3.4. Minimize queue count [38]
3.5. Minimize hop count [58]
3.6. Load balancing [15,33,87,88]
3.7. Minimize fragmentation [51, 92]
3.8. Minimize bandwidth loss [66,80–82,90,91]
3.9. Minimize GCL length [41,42,48,73]
3.10. Minimize topology cost [30]
2. Performance
2.1. Minimize worst-case for LTS [50]
2.2. Minimize latency [35,37,40,53,54,76–83,112,123]
2.3. Minimize jitter [78–81,116]
2.4. Maximize quality-of-control [84–86]
2.5. Minimize path length [53]
2.6. Application response time [54]
2.7. Balance traffic loss & latency [83]
4. Combinations
4.1. Wired & wireless networks [48]
4.2. Cross-domain scheduling [94]
4.3. Inter-flow dependency [47,49,93]
4.4. Combine shapers [11]
4.5. Combine TAS with preemption [95]
4.6. different technologies e.g. TSN, 5G [63]
4.7. Fog computing, VM migration [61,84,86,96]
5. Network Health
5.1. Resilience/ robustness [88,104–106]
5.2. Stability [29]
5.3. Safety/security [102,104,109,110,119]
5.4. Feasible schedule [14,39,60,63,69,97–100]
5.5. Reduce frame loss [83]
5.6 Fault-tolerance [30,101–103,120,126]
5.7. Multicasting [62,117]
5.8. Network model correction [39]

Figure 3. An overview of the objectives in the surveyed literature classified into aim groups, together with references to the corresponding papers. The objectives are enumerated for easier reference in the other parts of the paper.

4.1.1. Schedulability and Dynamism

The schedulability and dynamism of a network are crucial objectives, particularly for use cases that involve dynamic reconfiguration like fog computing. Improving the dynamism within the network, such as through dynamic reconfiguration, can be viewed as a means to enhance traffic schedulability. Consequently, various research works have focused on increasing both the schedulability of traffic and the dynamism of the network, employing different optimization objectives to achieve these goals.

For instance, one approach, introduced in [34], is the urgency-based scheduler (UBS), which supports scheduling of urgent traffic. Similarly, study [36] presented an alternative approach to address the same objective, suggesting enhancements to TAS (traffic-aware scheduling) and introducing a protection band. This band allows for the transmission of emergency traffic while isolating it from LTS and BE traffic. Another strategy to enhance network dynamism involves optimizing runtime scheduling. In this regard, Barzegaran et al. [61] proposed an extensibility-aware algorithm based on simulated annealing to maximize the schedulability of dynamic traffic. On the other hand, Syed et al. [59] formulated a mixed integer problem, leveraging the similarities between the scheduling problem and vector bin packing. However, both approaches ([59,61]) did not propose offline scheduling methods to prepare for schedule extension during runtime.

Another approach to optimizing runtime scheduling assumes the existence of a pre-configured schedule in the network and focuses on developing online reconfiguration algorithms. For example, study [65] utilized column generation to maximize the traffic acceptance rate, while study [66] introduced an incremental routing and scheduling approach to maximize the number of schedulable flows at runtime. In addition to maximizing online schedulable flows, several approaches aim to minimize reconfiguration time. Study [67] presented a joint-ILP formulation and a scheduling compatibility heuristic, while study [68] introduced HERMES, a heuristic for reducing reconfiguration time.

Another avenue to enhance network dynamism is through fast scheduling. Approaches that accelerate schedule synthesis include simplifying topology [69], incremental scheduling using divide and conquer [70], leveraging divisibility theory [33], partial scheduling [71], exploring the correlation between periods [72], and optimizing routing to expedite scheduling [73]. The interdependence between scheduling and routing has also been considered. For instance, authors in [47] employed a genetic algorithm where the chromosomes represent choices for routing along with application bending. In contrast, study [15] formulated an ILP to minimize the maximum load in a port, routing HTS traffic through different ports to reduce conflicts, particularly in larger networks. Additionally, Authors of [71] utilized a partial scheduling approach for space probing guided by conflicts, proposing EPIC, a heuristic for fast network scheduling in larger networks.

Various techniques have been employed to maximize schedulability. For instance, study [60] proposed several scheduling heuristics based on bin packing, dot product, coefficient of variation, and other methods. The authors of [74] focused on the trade-off between schedulability and optimality of solutions, using a constraint programming formalism. In [75], authors suggested a real-time routing scheduler to ensure the schedulability of HTS streams, while authors of [32] utilized an injection time planning algorithm to maximize the scheduling of HTS streams. Furthermore, some approaches have suggested enhancements in TSN hardware to improve schedulability. However, not only pure HTS schedulability has been considered. For example, a GRASP metaheuristic is developed in [45,46] to enhance the chances of AVB traffic while serving the HTS traffic. Alternatively, study [22] discussed the impact of temporal distribution on schedulability by altering the inter-frame time (i.e., sparsity of traffic) in three different setups. In contrast to [22,51,58], they argued that minimizing the window count or traffic sparsity on a link provides more opportunities for other traffic types and yields better scheduling results.

4.1.2. Network Performance

The performance of a network in terms of providing bounded latency, jitter, quality of control (QoC), and the mutual influence between different traffic types plays a critical role in delivering the network's QoS objectives. Consequently, numerous research works have focused on optimizing network performance, with minimizing latency being the primary objective in this category.

For instance, Studies [37,40,76,77] focused on minimizing latency for HTS streams. The authors of [37,40] employed exact methods based on ILP, while the authors of [76] developed a two-steps mechanism utilizing a hybrid genetic algorithm to find the schedule, which is subsequently compressed in the second step. Many works combine latency minimization with other optimization objectives. For example, works [78,79] aimed to minimize both latency and jitter. Craciunas et al. [78] adopted a first-order logic approach for incremental scheduling and utilized, in [78,79], an SMT solver to find the optimal solution. Similarly, studies [80,81] considered latency and jitter while aiming to minimize bandwidth utilization. They devised a genetic algorithm to rapidly calculate a good solution.

Other approaches strive to minimize latency alongside the number of guard-bands using deep reinforcement learning [82], balance latency and traffic loss using machine learning and genetic algorithms (e.g., NSGA-II) [83], minimize latency and application response time through application-level scheduling [54], and minimize path length using an ant colony optimization approach combined with a binary multi-knapsack formulation [53]. In addition to focusing on HTS traffic, some works consider LTS traffic as well. For example, work [50] aimed to minimize the worst-case scenario for AVB traffic by managing the HTS queue count and routing of AVB using a GRASP metaheuristic.

Furthermore, various approaches proposed to maximize the quality of control. For instance, authors of [84] deployed a constraint and logic programming approach to systematically search for optimal solutions, while in [85], a codesign optimization problem was formulated to find the optimal period that minimizes the settling time of the configuration. On the other hand, authors of [86] utilized a Tabu-search metaheuristic to increase diversification in a fog environment.

4.1.3. Efficient Resource Management

Resource management is a traditional and crucial optimization objective in computer networking, including Time-Sensitive Networking (TSN). Among the key objectives in this domain, load balancing and minimizing bandwidth loss take precedence.

Several studies ([15,87–89]) have aimed to address the joint scheduling and routing problem, considering that load balancing involves path selection. In the case of [88], an incremental algorithm was employed to schedule flows using an iterative local search approach, prioritizing an as-early-as-possible strategy. To determine paths, a greedy algorithm for multiflow routing was utilized. Another study, ref. [87], focused on addressing the out-of-order problem by employing a multipath mix-flow scheduling heuristic, which improved link utilization and load balancing concurrently. While most works focused on load balancing through routing, study [33] approached scheduling within the context of CQF shaping, utilizing a flow sequence analysis and an incremental scheduling algorithm (FLJ-VB), which minimized runtime while leveraging divisibility theory for load balancing.

Furthermore, various works considered minimizing bandwidth loss. For instance, work [82] employed reinforcement learning to minimize guard-band usage in a no-waiting scheduling algorithm. Ref. [66] employed an incremental routing and scheduling approach, complemented by a prerouting algorithm, to minimize bandwidth waste. Genetic algorithms ([80,81]) and constraint programming ([90,91]) were also utilized in different approaches.

Bandwidth waste often occurs due to suboptimal transmission window placement, leading to fragmentation. To address this, authors of [92] proposed minimizing the number of gaps between events to reduce space-time fragmentation. Another closely related objective is the minimization of scheduled events (GCL length in Figure 3). Ref. [41] adopted the first-order theory of arrays to formulate a window-based scheduling problem, then employed an SMT/OMT solver to infer solutions that minimized the number of GCL events. In the context of wireless networks, study [48] proposed an incremental approach based on segment fragmentation to generate a compact schedule while pursuing the same objective.

Other objectives include minimizing the cost of the network topology (e.g., [30]) through GRASP metaheuristics and constraint programming-based formulations, reducing queue usage by extracting and adapting to traffic properties (e.g., [44]), and minimizing queue count for HTS traffic (e.g., [38]). Assigning an excessive number of queues to HTS traffic leads to resource waste, particularly if no HTS traffic is present. Hence, it is crucial to allocate only the necessary number of queues to HTS traffic.

4.1.4. Combinations

Different use cases may require hybrid networks where more than one technology/option coexist, simultaneously. The main target of the works in this category is to study different options. Refs. [47,49,93] focused on the interstream dependency. This is usually the case in service chaining when the sending of a stream depends on the reception of another stream. While study [49] used a heuristic list scheduler, study [47] applied a genetic algorithm that combines routing, and task binding. Ref. [94] normalized the cycle between CQF and CSQF to solve the cycle, queue, and bandwidth mismatch. The combination of TAS and CQF was deployed in [11] to support different traffic types on the network. The authors started by selecting an appropriate cycle and scheduling unit, then injected the streams that had similar cycles together to avoid computational complexity. Other combinations are TAS with preemption for enhancing schedulability [95], combining TSN and 5G using constraint programming [63], and TSN for fog computing with VM migration. Ref. [96] suggested a rescheduling algorithm based on Breadth-First Search (BFS), and Ref. [57] proposed a configuration agent managing runtime reconfiguration using a heuristic based on list scheduling. Although some works have started to appear more recently, this area of TSN is still very much unexplored.

4.1.5. Network Health

The works in this category concern the overall functioning of the network with less focus on the achievable QoS. For example, studies [14,97–100] only aimed to find a feasible schedule that satisfies constraints. Other works focused on generating feasible solutions that are tolerant to network faults. Fault tolerance could be enabled using different techniques. Refs. [101,102] used redundant transmissions as a fault tolerance technique. The authors of [101] based their work on frame replication, while [102] adopted a proactive approach by calculating previously the number of transmissions per HTS stream instance. They also provided some fault tolerance for LTS and BE traffic in case of failures. Ref. [103] also used time redundancy. Although their technique increases the latency of the streams, it could be beneficial in case it is combined with space redundancy. Time redundancy also suffers from the overprovisioning of timing resources in the network. Ref. [104] aimed at maximizing the resilience of control applications as well as the minimum security level. Ref. [105] discussed the loss-of-synchronization and introduced a set of constraints to mitigate the effect of an out-of-sync event for a maximum resynchronization interval. Reliability awareness was discussed in [105,106]. Ref. [106] used an SMT formulation based on path redundancy. Ref. [39] suggested a network model correction approach based on automatic inference of TSN rules and constraints. Ref. [29] tried to guarantee the worst-case stability of the control application by considering the effect of delay and jitter on QoC. Ref. [107] discussed the trade-off between frame loss during network updates and suggested a scheduler that allows offline and online configuration without extra overhead. Finally, [62] suggested a flow fragmentation approach based on SMT to generate feasible solutions for multicast streams.

4.2. Literature Analysis

Our literature review on the scheduling problem encompasses more than 100 papers that were published between 2016 and November 2022. For a better overview, we classified them in Tables A1 and A2. As can be seen, the number of works mostly increased every year, showing an increasing interest in the research community to solve this problem. An important driver of this interest is the need by the industry to develop solutions for their use cases that can be deployed in real networks as TSN standards became more mature over the recent years and products started to evolve.

4.2.1. Diversity of Application Areas

Table 4 shows a possible mapping between the different application areas of TSN and the algorithms presented in Tables A1 and A2. For example, [108] suggested a routing and scheduling scheme for industrial IoT in the field of underground mining. However, besides industrial automation, the largest number of approaches was developed in the automotive field. Although TSN scheduling research in the automotive started early (e.g., [34] published in 2016), the work continued to evolve. Works from later years treated important aspects such as security and authentication [109], functional safety with regards to Automotive Safety Integrity Level (ASIL) standard [110], reliability [106], fault tolerance [110], and cross-domain optimization [59]. We believe the reason resides in the fact that the automotive network topology is well established and pretty static concerning the streams. Therefore, researchers focus more on guaranteeing the good functioning of the network rather than agility of communication. However, we expect this view to change since future vehicles are meant to have more communication with the outside world (i.e., other vehicles or infrastructure) especially in the context of autonomous driving. In addition, the in-car services are expanding with regards to infotainment, for example.

Table 4. Scheduling algorithm and TSN profile per application area.

Application Area	TSN Profile	Applicable Algorithms
5Gs Fronthaul/Xhaul	IEEE802.1CM/de	[63]
Industrial automation	IEC/IEEE 60802	All with few exceptions
Automotive	IEEE P802.1DG	[29,30,34,51,59,60,80,81,95,99, 101,104,106,110,111]
Aerospace	IEEE P802.1DP/SAE AS6675	None
Service provider	P802.1DF	
Audio video bridging	802.1BA	

4.2.2. Diversity of Approaches

Early works were mostly inspired by related fields such as TT-Ethernet and adapted solutions to make them work for TSN by, e.g., changing problem formulation constraints. More recently, solution approaches are becoming more diverse in applied techniques to solve the problem as well as in the use cases where TSN is applied. Figure 4 shows the number of related works on the y-axis for each given year. As can be seen, early works used mostly ILP and SMT problem formulations. However, starting in 2018, the number of heuristics and metaheuristics started to increase as researchers have been obtaining a better understanding of the problem. Approximative solution approaches (according to Figure 2) are interesting because they can provide reasonably good solutions in a shorter time than exact methods. As we will discuss later, algorithm runtime is an important metric when evaluating a given approach.

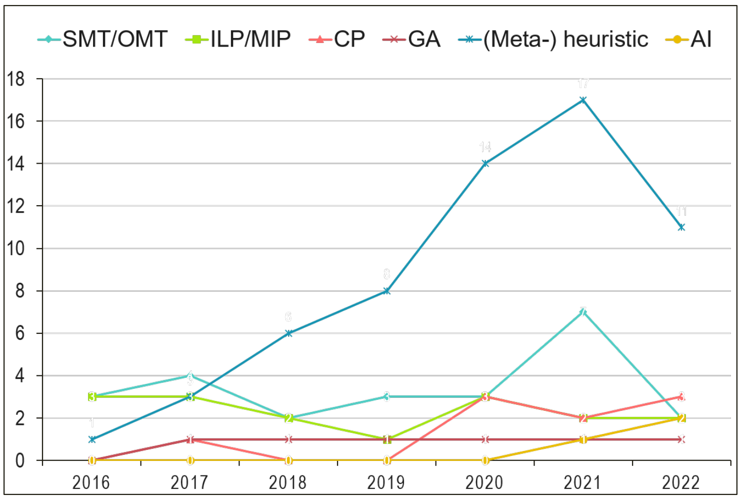


Figure 4. Evolution of approaches per year.

We also notice in the last two years the use of constraint programming (CP)- and artificial intelligence (AI)-based methods to solve the scheduling problem. Reinforcement learning (RL) specifically seems a promising approach as such an approach can automatically learn optimal policies that fit the network topology and stream characteristics. However, there are several challenges when using AI-based methods. First, dealing with hard constraints is still challenging when using, e.g., deep reinforcement learning. Second, finding solutions that are robust to deviations in the input (such as nondeterministic switch behavior or end-host jitter) complicates the problem further. Finally, generalizing solutions learned for one set of streams to a different scenario remains an open challenge. Here, recent works in the area of neural combinatorial optimization (NCO) techniques could be interesting candidates to find robust network-wide schedules that can deal with hard constraints.

4.2.3. Diversity of Objectives

In total, we identified 42 different objectives that different solution approaches aimed to achieve, which we summarized in Figure 3. More than 45% of the papers aim to achieve more than one objective, simultaneously. In around half of those cases, the objectives belong to different groups. The earliest and most commonly used objective is the minimization of latency (objective 2.2 in Figure 3), which appeared already in 2016, as can be seen from Figure 5 and continued to appear until 2022. In total, 15 papers that we surveyed considered it in their approach. Interestingly, more recent works shifted interest to also consider other aspects such as the dynamism of the network. We believe that this is again related to the maturity of TSN development. While in the early days, TSN researchers were looking more into proof of concepts, more recently there can be seen a shift towards practical deployment. As a consequence, the focus shifted more towards developing fast solution heuristics (i.e., objective 1.6 starting to appear around 2020 and was considered in seven works afterward, as seen in Figure 3) and considering domain-specific constraints such as resilience (objective 5.1 in the year 2019) or safety (objective 5.3 in the year 2019) related constraints. We also noticed an interest in scheduling optimization for fog computing after the year 2020. This includes aspects such as VM placement and migration.

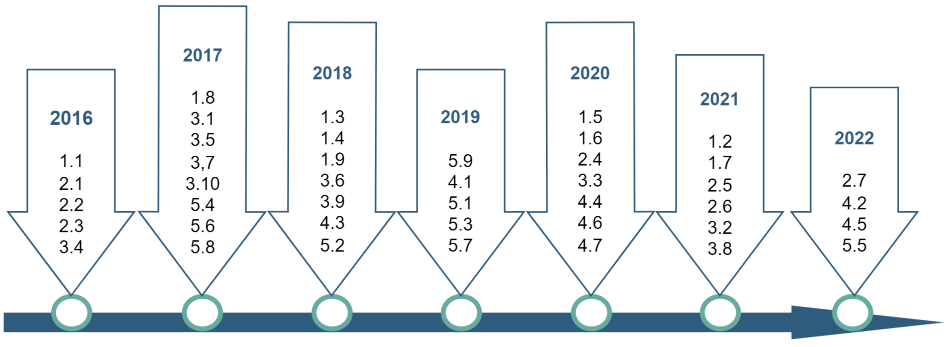


Figure 5. Evolution of optimization objectives over time.

4.2.4. Expansion of the Scope of the Problem

Traditionally, optimization of network configuration included only scheduling and routing. However, some of the more recent works included the topology design as well as the talker/listener placements in the optimization formulation, as can be seen from the third column of Tables A1 and A2 (T stands for topology and A for application (or task)). Few works so far include application placement optimization into the TSN scheduling approaches. However, with recent trends towards edge computing and TSN hardware support on commodity end-hosts (e.g., TAS-configurable hardware queues in the NIC) and virtualization (e.g., NFV and virtualized PLCs), flexible placement of TSN-enabled applications on the TSN-enabled edge-nodes is becoming a reality. However, to provide tight end-to-end timeliness guarantees and tight integration of end-host placement decisions, resource allocation in the end-hosts and their network stacks and a network-wide schedule configuration need to interact properly. On the other hand, a flexible placement of talkers and listeners on the TSN-enabled end-hosts adds another degree of freedom which could be exploited to increase the solution space of the scheduling problem and reduce the strictness of the requirement (i.e., placing HTS talker and listener closer in the network reduces the minimum required accumulated latency), and could lead to better robustness and less jitter by exploiting the topology design flexibility.

4.2.5. Diversity of Shapers/Use Case Specifications

Although earlier works focus on isochronous traffic, the later works consider other traffic types, too. The diversity in traffic specifications also includes considering multicast streams. Early works usually assume unicast streams, but later on some works [49,62,69,70,88,96,101,112] considered multicast streams. Other works tried to schedule sporadic traffic while imposing some constraints on the interframe arrival time. However, scheduling sporadic traffic is still challenging due to the criticality of the traffic in terms of latency while being irregular. Related to the diversity of the traffic specification, we noticed the increasing consideration of shapers other than TAS, although TAS is still dominant and most attractive. CQF is gaining momentum, especially in the context of multidomain optimization. We expect it to gain more and more interest since it tolerates bigger jitter than TAS while keeping the synchronous nature. Similarly, combining different shapers/shaping techniques is a very interesting direction of research as it has great potential. However, only a few works exist in this area, such as [95], which studies the combination of TAS with frame-preemption, and [113] which studies the combination of CBS with frame-preemption. Ref. [83] studied combining TAS with CBS and [11] studied combining TAS with CQF. It would be interesting, in this context, to study the effect of frame-preemption [12] on traffic shaping, which could improve the performance for urgent-high-priority frames further. Another interesting direction for future studies is to analyze the relation between the length of the tolerance interval and the slope accumulation rate as well as the rate of frame drops and the rate of frames missing their allocated windows.

4.3. Challenges for Network-Wide Scheduling

In this section, we discuss challenges that are inherent to network-wide scheduling. The questions we raise in this section are crucial to guarantee end-to-end QoS to the network streams, independently of their possible placement. Figure 6 summarizes the discussed challenges.

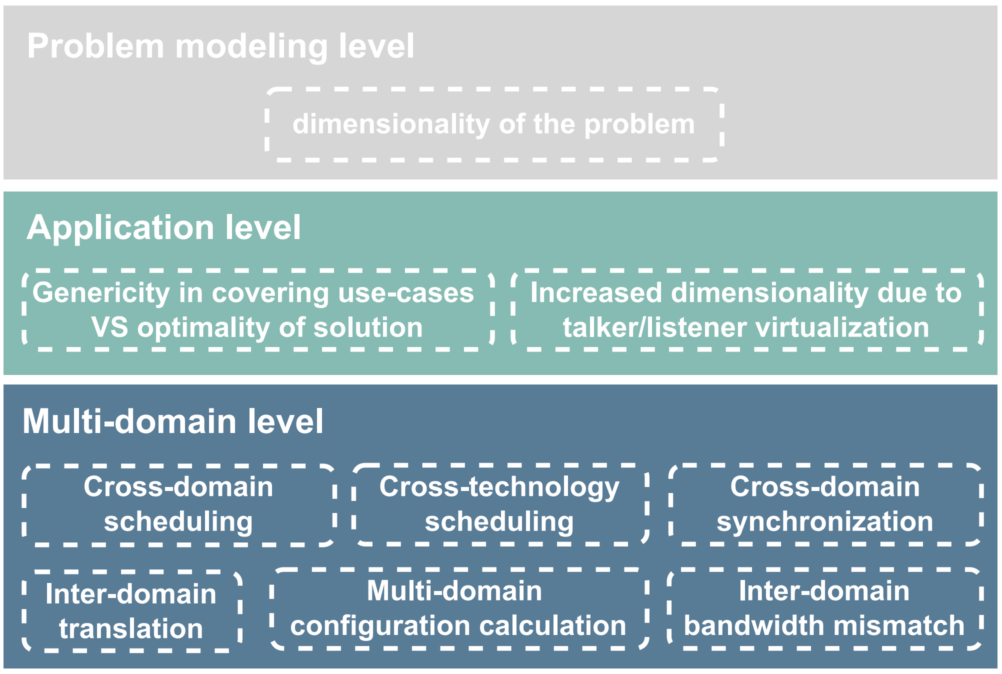

Problem modeling level
dimensionality of the problem
Application level
Genericity in covering use-cases VS optimality of solution
Increased dimensionality due to talker/listener virtualization
Multi-domain level
Cross-domain scheduling
Cross-technology scheduling
Cross-domain synchronization
Inter-domain translation
Multi-domain configuration calculation
Inter-domain bandwidth mismatch

Figure 6. Network-wide scheduling faces many challenges. They are mainly on the networking level.

- *Problem modeling level*:
  Regardless of the used methods to solve the problem, the dimensionality of the problem poses a challenge. In legacy Ethernet, forwarding traffic requires solving the routing problem and assigning flow priorities. This still holds in TSN networks but with the addition of the time dimension. As stated previously,there are generally two main approaches for scheduling: exact approach (i.e., mathematical formulation as an optimization problem with off-the-shelf solvers) or heuristic/metaheuristic design. Solving the network-wide scheduling problem has a high problem dimensionality due to the many constraints and decision variables when using exact methods. Ref. [8] shows that the number of constraints increases with the number of switches and quadratic with the number of frames. Therefore, large use cases with many streams to schedule may lead to excessive runtime of the solver in many cases (e.g., when using SMT solvers). One main problem is the existence of binary and/or integer variables that require combinatorial optimization methods to be applied. To reduce the dimensionality and, thus, the problem complexity, several works fix some of the variables, which reduces the runtime at the expense of solution optimality. For example, ref. [47] modeled the scheduling and routing problems jointly using a genetic algorithm. The challenge facing this approach is gene encoding. Solving routing with genetic algorithms consists of finding the mapping between a limited number of transmission windows and a limited number of ports. However, scheduling consists of finding the best instant to open the window, which could theoretically be anytime during the scheduling cycle. Therefore, encoding all possible times leads to an exploding size of the gene. Ref. [47] avoided this problem by finding the path using the genetic algorithm and inferring the schedule once the paths are fixed as the earliest time that the stream can traverse all the links in its path without interference.
- *Application level*:
  - Due to many mechanisms and standards, TSN is gaining attention from several industries with very divergent use cases. Each use case may require different mechanisms. Therefore, the standardization community is trying to define different profiles. A profile specifies the set of choices (i.e., features, default configuration values, etc.) to apply. Table 4 summarizes the current TSN profiles per application area. The divergence between the requirements of applications from different areas places the scheduling algorithms against the challenge of being generic enough to serve applications belonging to different areas while being specific enough to provide optimal schedules for all applications regardless of their divergent requirements.
  - A second challenge on the application level is related to the virtualization of end-hosts. With the introduction of virtual PLCs and NFV that could be deployed in a set of possible edge compute nodes having TSN stacks and hardware support, the end-host could be attached to different locations in the topology. This brings an additional degree of freedom that could increase the solution space but also complicates the scheduling further to the increased problem dimensionality.
- *Multi-domain level*:
  TSN Ethernet as a deterministic networking technology is not alone in the ultra-low latency ecosystem. Other technologies such as 5G, Deterministic Networks (DetNet), and wireless TSN are also gaining momentum. Industrial applications could involve any combination of these technologies in a large-scale hybrid network. Similar problems exist in a pure TSN setup where multiple domains are involved. One example presented in [59] is the in-vehicle network where electronics and mechanical systems are distributed in domains (e.g., autonomous driving domain, power train domain, chassis domain, body domain, telematics, and infotainment). Cross-domain functionalities need the collaboration of different domain controllers. This setting brings about more challenges related to hybrid multidomain scheduling.

- An important challenge to solve is a potential clock mismatch. Mechanisms such as TAS require network-wide synchronization. Technologies that support time synchronization such as 5G ease the problem slightly. Ref. [114] discusses the integration of TSN and 5G. Since 5G supports multiple time domains, it could join the TSN time domain by synchronizing one of its working clocks to the TSN master clock according to the gPTP procedure. However, this is not available in other scenarios such as having two TSN domains, each having its own controller and master clock. A possibility to cope with this challenge is to have a synchronization controller that resides at the border of all the domains and belongs to all of them. The synchronization controller could solve the problem by playing a GM role. Nonetheless, the synchronization controller would be a single point of failure. Another option is to introduce time awareness to the control plane such that the CNCs exchange synchronization information and consider the time difference while calculating the cross-domain schedule.
- A second challenge is related to bandwidth mismatch. In this case, the interdomain interface could be a congestion point. Ref. [94] suggests using network slicing and queue length management to mitigate this mismatch.
- A third challenge is related to the configuration generation for multidomain operation. The configuration could be generated centrally once the network-wide scheduling problem is solved by only one orchestrator and distributed to all network elements. Here, the other orchestrators should share their domain information. The challenge here is that different technologies have different features and a single orchestrator controlling all is not a viable solution. For example, the central network controller (CNC) from TSN does not support 5Gs control features such as network slicing, etc. The 3GPP suggested a solution in release 16/17 which exposes the 5Gs as a virtual switch to the CNC [114].
- The same logic could be applied in the general case of multidomain configuration. Suppose we have two domains, each controlled by a separate controller. In that case, we can introduce a higher-level CNC that sees each domain as a virtual switch. The domain controller must then expose the domain's resources as a virtual switch with certain capabilities. Afterward, the higher-level CNC calculates the global configuration and distributes it to controllers of the domains. Finally, each domain controller calculates and deploys the actual configurations such that it meets the configuration calculated by the higher CNC.
- The previous mode of functioning combines centralized and distributed computation of configuration. However, such a hybrid mode has other challenges related to scheduling time and the control of traffic overhead. A third approach is the distributed mode, where every controller calculates the configuration of its domain locally without consulting the other controllers. This approach still needs an entity to verify the end-to-end requirement. That could be the talker and/or the listener, with the help of all the domain controllers in between. In this case, a distributed cross-domain scheduling protocol is required. As for the cross-domain scheduling algorithms, only a few works exist. For example, ref. [94] suggested a platform for cross-domain coscheduling based on CQF and CSQF (Cycle-Specified Queueing and Forwarding). The authors suggested some solutions for cycle and bandwidth mismatch but still have open issues related to reliability by frame replication and elimination and sporadic traffic handling.
- A fourth challenge is related to the interface between the different domains. If domains involved in the forwarding of the frame are using different technologies, frames crossing the border between them need to be modified. Different technologies use different features and traffic formatting. For example, the priority of an Ethernet frame is encoded in the PCP field of the header, but for wireless communication, the packet header has a different structure (e.g., in the 5G system, the priority of a packet is encoded in the 5QI field). Therefore, a translator is

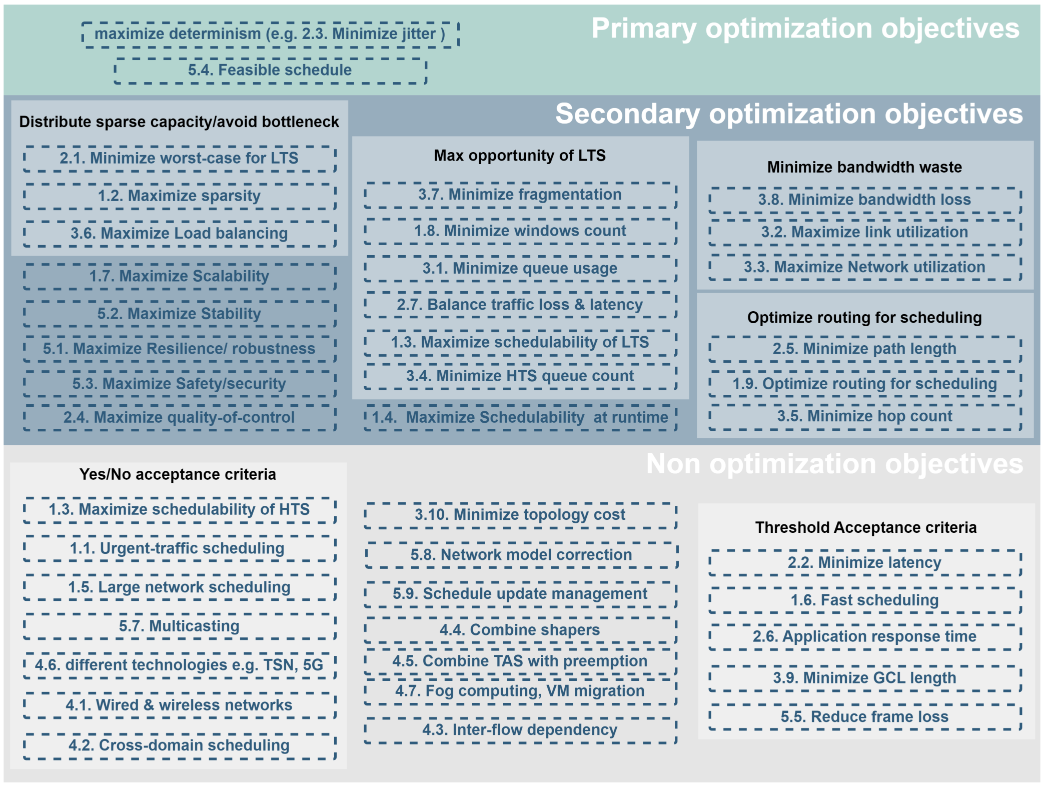
required that rewrites packet headers or encapsulates packets and translates the time synchronization traffic without precision degradation. Ref. [115] suggested a hybrid architecture for a TSN device that allows the integration of wire-line and wireless TSN. In this regard, one challenge for distributed cross-domain scheduling is to consider the latency added by the translation operations at the translator. If the schedule is calculated centrally, then the centralized controller should have the information about the translator. However, in a distributed mode, every domain controller considers only the ports belonging to it, but the translator partially belongs to different domains. This brings additional complexity to the network-wide scheduling.

5. TSN Scheduling Problem Formulation

A proper configuration of the network-wide TAS parameters requires the solving of a TSN scheduling optimization problem, which includes the definition of optimization objective(s), problem constraints, and model assumptions. The outcome of the problem is a proper configuration of the TSN elements (TAS parameters such as GCL and proper paths) to fulfill the required QoS of all scheduled streams. However, there are many different problem formulations proposed in the literature. For example, refs. [38,41,97] use similar sets of constraints. However, their optimization objectives are very different. Therefore, in this section, we discuss the main optimization goals, constraints, and model assumptions.

5.1. TSN Scheduling Optimization Objectives

In this section, we discuss the relevance of optimization objectives. We analyze the importance of those objectives and we classify them into three groups according to their relevance. Figure 7 summarizes our classification effort.



maximize determinism (e.g. 2.3. Minimize jitter)
5.4. Feasible schedule
Primary optimization objectives
Secondary optimization objectives
Distribute sparse capacity/avoid bottleneck
2.1. Minimize worst-case for LTS
1.2. Maximize sparsity
3.6. Maximize Load balancing
1.7. Maximize Scalability
5.2. Maximize Stability
5.1. Maximize Resilience/ robustness
5.3. Maximize Safety/security
2.4. Maximize quality-of-control
Max opportunity of LTS
3.7. Minimize fragmentation
1.8. Minimize windows count
3.1. Minimize queue usage
2.7. Balance traffic loss & latency
1.3. Maximize schedulability of LTS
3.4. Minimize HTS queue count
1.4. Maximize Schedulability at runtime
Minimize bandwidth waste
3.8. Minimize bandwidth loss
3.2. Maximize link utilization
3.3. Maximize Network utilization
Optimize routing for scheduling
2.5. Minimize path length
1.9. Optimize routing for scheduling
3.5. Minimize hop count
Non optimization objectives
Yes/No acceptance criteria
1.3. Maximize schedulability of HTS
1.1. Urgent-traffic scheduling
1.5. Large network scheduling
5.7. Multicasting
4.6. different technologies e.g. TSN, 5G
4.1. Wired & wireless networks
4.2. Cross-domain scheduling
3.10. Minimize topology cost
5.8. Network model correction
5.9. Schedule update management
4.4. Combine shapers
4.5. Combine TAS with preemption
4.7. Fog computing, VM migration
4.3. Inter-flow dependency
Threshold Acceptance criteria
2.2. Minimize latency
1.6. Fast scheduling
2.6. Application response time
3.9. Minimize GCL length
5.5. Reduce frame loss

Figure 7. Classification of optimization objectives by relevance. The optimization objectives in this figure use the same references as in Figure 3.

The first group concerns objectives that can be considered more as an acceptance criterion of the solution rather than an optimization objective. In the case that the solution satisfies the criteria, it can be used for the use case, and if not, then it is not convenient. For example, minimizing the end-to-end latency of the streams (e.g., [37]) is one of the earliest proposed objectives in the literature. Nevertheless, as long as the calculated schedule respects the delivery requirement of the stream (latency or deadline), the actual stream latency is not important, and consequently end-to-end latency can be seen as a constraint rather than an optimization objective. Other works such as [41] aim to minimize the schedule's length, which is typically between 8 and 1024 entries [41] to minimize the hardware resources used inside the switch. Unless the length of the schedule influences the performance of the forwarding behavior of the switch or minimizing the length of the schedule is targeting the applicability of the scheduling algorithm to more switches, this objective can, rather, be seen as a hardware constraint. Some works such as [58] aim to maximize the number of scheduled streams. However, for many industrial use cases, the number of HTS streams to be scheduled is given and has to be fully served. Consequently, this parameter can be seen as a hard constraint on the problem. On the other hand, maximizing the number of schedulable LTS streams might be important for some use cases, which require that the network is properly designed to provide enough resources. In this context, objectives such as the topology cost or the support of different techniques are not part of the optimization of the schedule, although they could be related to it. In Figure 7, we aim to cluster similar objectives. While some of them only require a simple yes/no question to answer (e.g., whether the solution algorithm supports multicast streams), others require the definition of a threshold (e.g., maximum allowed schedule synthesis time).

In summary, minimizing the schedule length, the end-to-end latency, or the number of scheduled HTS streams, among other objectives in this first group, should be modeled as constraints rather than being part of the objective function of the optimization problem.

Another group of objective functions can be classified as secondary as they aim to improve the performance of the overall system rather than guarantee the schedulability of the given HTS streams. For example, several works aim to distribute spare capacity and avoid future bottlenecks [59,60]. Indeed, keeping spare capacity on all links maximizes the success probability in case HTS streams need to be scheduled during runtime once existing streams are already placed. For example, ref. [59] aims to minimize the absolute deviation of individual link utilization, while [58,87] balances the load by minimizing the maximum load for all links. Another optimization objective is to maximize the opportunity for LTS traffic. Here, ref. [11] aims to minimize the average queuing delay for LTS streams by combining TAS and CQF. Ref. [24] aims to maximize the porosity of the schedule (the distance between HTS windows) to prevent back-to-back transmission of HTS streams and allow more BE traffic. Optimizing the assignment of queues to traffic classes is another important objective (e.g., ref. [38] aims to minimize the number of HTS queues).

Minimizing bandwidth waste is also an important optimization objective since bandwidth is a costly resource [18]. Bandwidth waste may happen for several reasons, e.g., if two HTS windows are scheduled after each other but the gap between the closure of the first and the opening of the second is so small that it does not allow the transmission of any LTS or BE frame. Bandwidth wastage could also happen due to frequent guard-bands. However, guard-bands, when required by the HTS streams, can not be avoided except between two streams of the same type that could be scheduled back to back without the need for a guard-band. In that case, precautions against jitter should be taken. Consequently, minimizing the number of guard-bands is not a relevant objective.

In summary, this second group of objectives is not crucial to model the schedulability of HTS streams, although it could be beneficial in specific scenarios.

One of the most important objectives of TSN is to provide guarantees in terms of bounded latency to streams, which requires eliminating nondeterminism from the network mainly caused by variable queuing delay leading to high jitter. Different aspects contribute to nondeterminism including synchronization errors, imprecise bridge delays considered during scheduling or frequent configuration updates, and the inconsistency that could bring to the network. While it is difficult to eliminate all of those problems, determinism can be improved by minimizing jitter. This could be performed at every hop inside the delivery path or at the last hop by making sure that the worst-case latency of an HTS stream should be smaller than the required maximum latency. Managing the jitter at the last hop could be achieved by making sure that the stream is always delivered with the worst-case latency by increasing its queuing delay at the last hop when needed (e.g., by exclusively allocating a queue at the last hop as in [116]). Consequently, the management of the queues becomes easier. However, the complexity of this approach is expected to increase with the number of HTS streams in the network since the jitter is not controlled in the network except for the last hop, which also would negatively impact the worst-case latency for all streams.

The main goal of this discussion is the importance of determinism for the TSN scheduling problem. Only five papers (see Figure 3) of the analyzed literature treated jitter minimization as a central objective of their work. This could be explained by the fact that many use cases do not need a very high level of determinism and that they can tolerate, to a certain extent, some randomness which renders jitter minimization not very relevant. However, if the use case does not require determinism, then it would be more convenient (i.e., less configuration effort without performance loss) to use CQF than TAS. Nevertheless, Tables A1 and A2 show that only seven papers use shapers other than TAS.

In conclusion, we believe the choice of the right shaper is very important and that in the case that TAS is selected, maximizing determinism should be the primary objective of the work. Nevertheless, this primary objective could be combined with other objectives, or constraints could be added that guarantee the feasibility of the solution.

5.2. TSN Scheduling Problem Constraints

An important part of the TSN scheduling problem formulation is to identify constraints that limit the search space to only the feasible solutions. Optimally, the search space is equal to the solution space, i.e., every viable assignment fulfills all the constraints. Consequently, reducing the number of constraints eases the finding of an optimal solution. Nevertheless, different works use different constraints that are convenient for their formulation. In this section, we revisit the constraints used in the surveyed literature, classify them, and aim to analyze their coexistence in the different works. Although the specific set of constraints depends on the problem formulation, we can identify the following classes:

1. **Constraints on the streams:** This set of constraints aims to look at the interaction between the streams and the network. From one side, each stream individually has a set of requirements from the network (i.e., requirements on latency, jitter, periodicity, and reliability) and from another side, all the streams impose more requirements collectively (i.e., dependency between streams, and sharing of the network resources considering all the streams).
    (a) MaxLatency constraint [11,14,24,31–33,35,38,40–42,44,46–50,54,60,62,63,65,69,70,72,74,76–79,82,84,86–88,90,92,94–98,101,105–107,109–112,116–121]: This constraint was used in most of the works and limits the allowed end-to-end latency of the streams.
    (b) MaxJitter constraint [11,41,62,63,68,79,84,90,111,116]: This limits the allowed jitter of the streams.
    (c) Periodicity constraint [32,38,41,48–50,54,59,60,62,67,70,72,76,78,79,82,87,91,97,98,101,102,107,111,116,120]: This ensures that the full transmission is happening during the stream cycle, taking into consideration all the network delays (i.e., processing delay, propagation delay, transmission delay, and queuing delay)
    (d) Slot constraint [11,32,58,82,97]: This defines the size of a time slot if the problem formulation discretizes the time (e.g., with CQF).
    (e) Reliability constraint [88,106]: This focuses on the reliability of the streams and is used specifically in the case of redundant/multicast transmission.

   (f) Interstream dependency constraint [47,49,54,86,109,119,121]: This is required in case application embedding in the end-hosts is considered a part of the scheduling problem. This constraint is useful for use cases that involve service chaining.

2. **Constraints on the queues:** This set of constraints aims to look at the configuration of the queues in order to enforce a certain queuing discipline.
   (a) Allocation constraint [41,74,97,116,118,121,122]: This ensures that the traffic classes are properly allocated in the right priority queue (e.g., exclusive queue allocation introduced in [116]).
   (b) Accessibility constraint [24,31,35,38,41,42,44,46,50,58,71,74,79,82,84,86,87,92,97,102,105,109,116–120,122]: This controls the order of frames in the queues or limits the access of the different frames/flows to a certain queue. This constraint is also called isolation constraint in the literature and it has different variations, e.g., frame isolation, flow isolation [38], and size-based isolation [116].
   (c) Zero-queueing constraint [59,72,98,117]: This forbids the HTS frames to be queued in order to minimize latency and/or jitter.
3. **Constraints on the frames:** This set of constraints aims to look at frame-related aspects concerning its specifications (e.g., size), its relation with other frames (e.g., sequencing of frames), and its relation with the network (e.g., path).
   (a) Sequencing constraints [24,72,78,87,95,98,116,121,122]: These ensure that reordering of frames belonging to same stream will not occur in the network.
   (b) MaxSize constraint [37,79,84,123]: This limits the maximum allowed size of the frames.
   (c) Routing constraints [14,29,30,37,38,40,42,47,49,50,58–60,67,69,70,72,74,78,79,84,87,98,101,104,106,109,110,112,119–122]: This constraint is used to enforce certain routing rules on the frames, e.g., to prevent loops in the path, to limit the number of hops in the path, or to ensure the frames belonging to the same stream follow the same path. This constraint could be applied to scheduling problem with fixed paths.
4. **Constraints on the transmission windows**: This set of constraints aims to look at the transmission of frames in the link. It concerns the start and the end of transmission, and the spacing between consecutive transmissions.
   (a) Size constraint [11,24,33,35,41,42,53,58–60,63,65,79,91,92,94,95,97,100,101,106,109,111,119]: This limits the size of the windows, the amount of tolerance, the maximum occupancy of a link (e.g., by limiting the maximum number of windows in a link during one cycle for example), etc.
   (b) Offset constraint [14,24,29,31–33,41,42,44,46,48,54,59,62,63,66,69–71,74,76,78,84,86,90–92,95–97,102,104,105,107,111,112,117–122]: This controls the absolute positioning of the windows in relation to the gating cycle. This constraint ensures that the transmission of a frame is planned only after its full reception (i.e., enforce store-and-forward mode of functioning).
   (c) Overlap constraint [14,29,31,35,37,38,40–42,44,46–50,53,54,58,63,66,67,69–72,74,76–79,84,86–88,90–92,95–98,102,104,106,107,110,111,116–122]: This ensures the temporal isolation between frames on link usage and prevents the double assignment of a transmission window to a frame.
   (d) Spacing constraint [24,41,48,59,62,98]: This limits the spacing between windows of the same stream or different streams. This could be due to hardware limitations (as in [41]).
5. **Constraints on the resources** [11,31,32,42,48,49,54,63,67,69,72,78,96,97]: Some works additionally consider resource constraints related to limiting the length of the schedule or the application memory and CPU requirements if application placement is regarded (e.g., ref. [78]).
6. **Constraints specific to the problem** [29,30,47,104,110]:
   Some of the constraints in the reviewed literature are very dependent on the research

question treated by the respective approach. Those constraints are not related to the scheduling problem itself; rather, they are needed to find feasible solutions. An example of such constraints are the ones used to comply with the functional safety standards, such as ASIL reqs constraints, as in [110], or the stability constraint used in [29] to enforce a stability margin in the solution. Other works treating application embedding introduced a constraint for the unique bending of the application in the end-hosts. Topology constraints are also considered problem-specific, as optimizing the topology is a step prior to its configuration.

The constraints presented above have different importance, which is reflected by how often they are used in the reviewed papers. Some constraints are very crucial to the scheduling problem formulation itself, while others are less relevant. To illustrate the relevance of the different constraints and to understand the interaction between them in more detail, we illustrate in Figure 8 a network graph showing the relation between the different constraints. In the figure, the nodes represent the constraints, and an edge connecting two constraints represents how often they are used together in the same work. Figure 8 identifies a subset of constraints that are central to the scheduling problem and classifies the rest as peripheral. We note the repeated coexistence of the maximum latency constraint, the overlap constraint, and the offset constraint as very crucial in the formulation of the scheduling problem. The accessibility constraint, the routing constraint, and the periodicity constraint occur less often. This illustrates that the focus on achieving a feasible solution with smaller latencies was more important than achieving a higher level of determinism. Determinism could be enhanced using the accessibility constraint or the maximum jitter constraint, which were not very present in the works. The remaining constraints, other than these six, are less important, although useful.

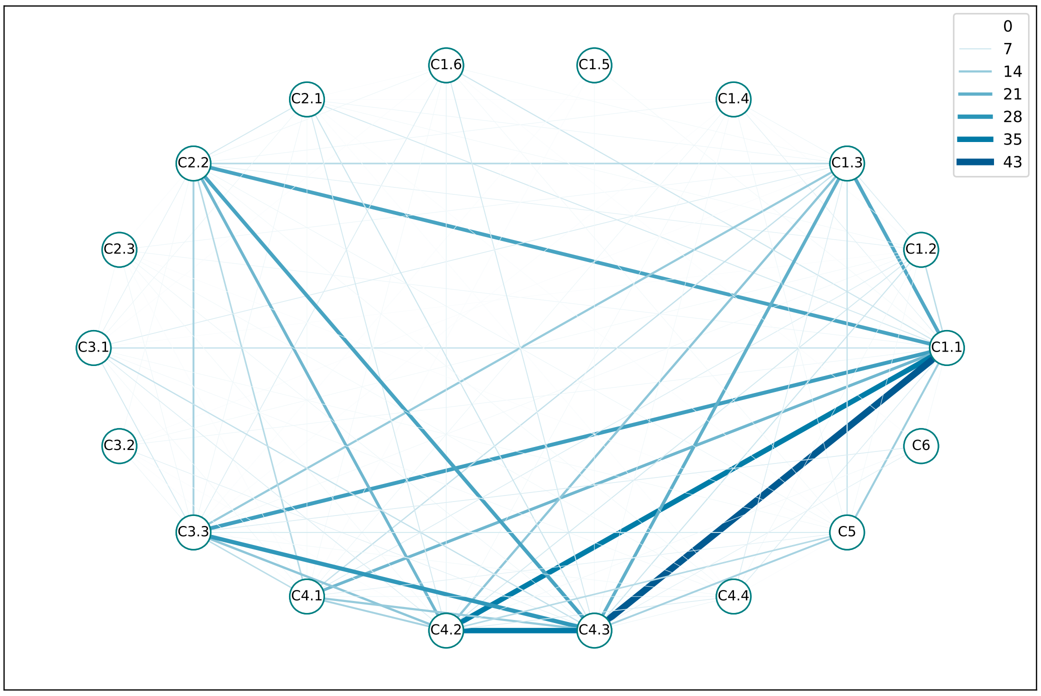


Figure 8. Correlation between the occurrence of constraints in the reviewed literature.

5.3. TSN Scheduling Problem Formulation Assumptions

Assumptions are a crucial part of the scheduling problem formulation. Practitioners use them mainly to simplify the problem formulation conceptually and/or computationally or to define the applicability scope of the work with regard to the topology, the resources of the network, or the use cases. However, many assumptions may not be realistic or may not reflect reality and, thus, may render a once-found solution not useful in practice.

The assumptions discussed in this section can be classified broadly into two categories. The first category concerns the necessary assumptions, such as synchronization for TAS or defining the input of the problem. We notice that most of the works do not explicitly mention them. Although some of those assumptions seem trivial, they are still necessary for the validity of the problem formulation. The second category of assumptions is those expressing the choices of the authors without being explicitly required by the problem itself. Authors use this type of assumption for mainly two reasons. The first reason is to ease the problem formulation without losing generality. In this case, the assumptions should be properly justified. The second reason is to limit the applicability of the solution to specific use cases (i.e., use cases with certain requirements on the traffic) or networks with specific characteristics (i.e., networks with certain capabilities, hardware, and/or topology).

5.3.1. Assumptions on Synchronization

A common view of the time is required for TAS to function properly. Therefore, TAS scheduling can not happen without assuming global synchronization. While the authors of [11,15,24,31,41,47,49,65,67–70,72–74,94,105,116–119,123] assumed a globally synchronized network, this assumption is not required for asynchronous traffic shapers such as ATS or CBS. Ideally, time synchronization has no error [68], but in real networks, synchronization imprecision is inevitable. For example, [46] assumed a network precision of 5.008 µs, and [105] assumed a maximum clock deviation between nodes after an out-of-sync event to be 1200 µs which are more realistic assumptions.

5.3.2. Assumptions on the Queues

The management of the queues, although not part of the shaping, is very crucial for the problem formulation. Different works try to reduce the complexity of the scheduling problem by assuming a specific queue setup. For example, [11,36,38,68] attributes the highest priority to the HTS queues. As the mapping between the traffic classes and priority queues is configurable, assigning the highest priority to HTS queues is a reasonable choice. Other works made different assumptions on the queues, such as fixing the number of HTS queues to one (as in [46,47,49,57,79,122]) or preventing the opening of more than one queue gates at the same time (as in [50]). Assuming the highest priority for the HTS queue could be used to prioritize HTS traffic when multiple queues are eligible for transmission. However, limiting the number of queues with an open gate at the time to one marginalizes the priority of the queues since only one would be always eligible for transmission. Therefore, combining the former two assumptions is not beneficial.

5.3.3. Assumptions on the Forwarding Mode

Many TSN switches support two different forwarding modes. In the store-and-forward mode, the switch processes a frame only after its full reception. In the cut-through mode, the switch starts processing the frame after only the necessary information is received (i.e., its header). Refs. [34,37,74] assumed that all switches are in store-and-forward mode while [40,112] assumed the switch delay to be 4 µs, which is equivalent to a cut-through mode.

5.3.4. Assumptions on the Cycles of the Streams

Some of the trivial assumptions found in the literature are that all flows are cyclic, as in [120], or that the cycles of all the streams, among other stream information, are known precisely, as in [66]. Those assumptions are not explicitly mentioned in the rest of the works because, as shown in Table 3, TAS is mainly needed for HTS traffic which is by definition cyclic, and the streams information, including the cycles, is necessary to calculate a schedule. However, several assumptions are made regarding the cycle itself. Refs. [24,37,58,72,79,86] assumed that all the streams have the same cycle. This assumption reduces the length of the GCL since it is built for a single cycle (equal to the cycle of all the streams) without the need to check the overlap of the frames' transmission in the subsequent cycles. Nevertheless, this assumption limits the applicability of the TSN scheduling algorithm for specific use cases, as in general, different flows may have different cycles. Ref. [124] aimed to reduce the length of the GCL differently by assuming that the cycles of the streams are all multiples of the GCL cycle. Another type of assumption is relating the cycles of the streams to their deadlines. For example, the authors of [24,50,102,103] assumed that the deadlines of the streams are equal to their cycles, and the authors of [39,74] assumed that the deadlines of the streams are smaller than their cycles. Although these assumptions are not always valid, authors use them to avoid the coexistence of different instances of the same stream in the network, as that could be a reason for intrastream overlapping.

5.3.5. Assumptions on the Delays of the Network

A simple delay model considers four types of delay: (i) the processing delay of a frame in a node, (ii) the queuing delay of a frame in a priority queue, (iii) the transmission delay of a frame from the queue to the medium, and (iv) the propagation delay of a frame on the link. Although this model is vastly adopted in the literature, there is no consensus on the actual delay values. Refs. [40,50,51,65,88,98,104,112] considered the processing, transmission, and/or propagation delays constant, [37,48,73,82,98,112] assumed them to be the same in all the switches, [47] considered the processing delay to be relative to the frame size, [49,82,104] assumed different values for the transmission delay ranging from 1Mbps to 1Gbps, [46,70,91,120] ignored the propagation delay or considered it as part of the network sync error, and [49,59,60,76,80] considered a zero queuing delay. As for the delay in the end-hosts, [121] ignored the delay in the end-host, and [104] fixed it to 0.5 ms. This shows the need for a more detailed study that provides a better understanding of different delay components in TSN networks. For reference, according to IEEE802.1Qcc [125], the bridge delay is defined as a tuple of (ingress port, egress port, traffic class). For every tuple, there are two types of delay: dependent on the traffic, and independent. Every type is represented by an interval that could be read from the bridge. However, the exact values depend on the configuration. Ref. [125] recommends reading these variables after configuring the bridge to obtain the most accurate values. Therefore, a better option is to consider the bridge delay as an interval and make realistic assumptions on interval boundaries.

5.3.6. Assumptions on Preemption

Frame-preemption is an important technique to deal with sporadic and high-priority traffic. However, not all works consider this type of traffic. Refs. [16,24,34,103,123] assumed that there is no preemption. Since frame-preemption could be activated or deactivated as needed, this assumption may be required for some use cases. For example, if the MTU is small and the transmission speed of the link is high such that the latency accumulated by the HTS streams does not violate its maximum latency constraint, then frame-preemption might not be necessary. In this case, the low-priority stream could finish its transmission in order to avoid the extra complication brought to the network configuration by frame-preemption. Nevertheless, the authors could choose to consider preemption in order to have more flexibility in the schedule. Ref. [50] assumed that only HTS traffic could preempt AVB traffic, and [120] considered all time-triggered traffic preemptable.

5.3.7. Assumptions on the Hardware

Assumptions on the hardware include assumptions on switches, links, and end-hosts. As for the switches, [119] required all nodes to be TSN-capable, [48,86,123] assumed identical switches in the network, and [48,78] were more precise and assumed identical memory and fixed buffer size. As for the links, [14,72,126] required all links in the network to be identical, [49,117] assumed all links to be full-duplex, and [34] assumed that there is no overprovisioning of the links. More specific assumptions were discussed in [75], where 25% of the link bandwidth was assumed to be used by BE traffic, and [41], which established an equivalence between egress ports and links connected to them. This assumption is realistic for wired communication since every port can be connected only to one link. However, for wireless communication, this assumption does not hold. Ref. [48] instead assumed a given number of replicas on wireless links. Finally, [61] made multiple assumptions on the fog end-host such as limiting the number of CPU cores to one.

5.3.8. Assumptions on Traffic Characteristics

Focusing on the HTS traffic, [119,120] assumed a maximum size of the frames (e.g., MTU = 1500 bytes), and [41,73,74,87,118] assumed that every stream has only one frame per cycle. Such assumptions may lead to multiple problems if the talker does not perfectly arrange the sending of the frames . Any gap between frames could result in the last frame always missing the allotted time slot. One way to avoid this issue is to control the sending offset of the frames. Refs. [71,91] required a zero offset at the talker and [41] assumed a controllable offset. In general, legacy talkers do not support TSN mechanisms (e.g., off-the-shelf sensors). Consequently, this assumption is applicable only to specific use cases where talkers support controlling transmission offsets. Nevertheless, for sporadic traffic, controlling the offset is not possible. Refs. [22,113] assumed a minimum time gap between two consecutive transmissions of sporadic traffic. This assumption is very specific to the use case (e.g., the emergency red button is sporadic and the button could be pressed at any time). However, it could be justified in some scenarios. For example, in the case of interstream dependency caused by tasks chaining, the task should be given enough time to process an incoming stream before sending its response; [78] assumed that talkers send and receive only at the end of the task. Other assumptions include considering only unicast streams, as assumed in [41,51,72,76,84,86,102]. While this may be true for some specific use cases, it may lead to suboptimal solutions in the presence of multicast streams. Multicast streams sourcing from one talker need to traverse a switch where frames need to be replicated on multiple ports for forwarding along a multicast tree. Casting multicast stream allocation to unicast streams may result in additional resource reservation or inconsistency in network configuration.

5.3.9. Assumptions on the Problem Design

Many works used assumptions on the input of the problem. Examples are assuming that the topology is fixed (as in [45,46,117]), the routes are given (as in [41,63,69,86,116,118]), and the positions of the talkers and listeners are fixed (as in [79]). These assumptions eliminate the possible extensions of the scheduling task such as joint routing, topology optimization, and application embedding. On the contrary, other works used assumptions from this group to extend the problem even further by considering redundancy aspects, such as [30,109] assuming that the redundancy level is given or [126] assuming the tolerance level (i.e., the number of disjoint paths) is provided. In the same way, [104] considered security aspects by assuming that the security importance is an input of the problem and establishing a set of assumptions on the attacker. Refs. [88,102,103,120] considered fault tolerance by introducing some assumptions on the fault model such as assuming single faults with minimum interarrival time, assuming a fault affects only one frame, assuming a single fault in the cyclic redundancy check (CRC), or assuming a constant error rate.

Another type of assumption related to the problem design touches the formulation of the problem. For example, assuming the use of TAS, as in [51], assuming an ideal network (no loss, no drop, error-free channel), as in [85], assuming valid solutions always meet the deadline, as in [117], assuming a densely connected topology where path always exists, as in [117], assuming certain characteristics on the path of the traffic such as there is at least one switch in the path, as in [86], the path of the stream does not change from one cycle to another, as in [110], and there are no loops in the paths, as in [95]. Although these assumptions ease the problem formulation to some extent, they have an applicability issue. A clear example is the consideration of an ideal network. This is physically not possible and, consequently, the algorithms built for an ideal network are not usable in real use cases. Other assumptions from this group include considering that maximizing gaps between HTS windows reduces starvation of other traffic types, as in [52]. Although this makes sense intuitively, the assumption itself needs formal proof to consolidate it.

Likewise, [99] assumed that the traffic is always schedulable. This assumption depends on the resources available in the network as well as their distribution. For example, if bandwidth is highly fragmented, then it might happen that the total available bandwidth is enough to accommodate traffic, but in reality, there is no fitting slot. In the general case, this assumption is not valid. Assuming that the maximum path length is 8 hops (as in [37]), this assumption may hold for specific cases, but in general, paths can be much longer (e.g., industrial automation profile [10] supports up to 64 hops).

6. Metrics for TSN Schedule Evaluation

The TSN scheduling problem can be solved by different approaches, such as using a standard solver (e.g., Z3 used in [62]) that aims to find an optimal solution that satisfies the constraints. Some approaches just aim to find feasible solutions that satisfy the constraints. Alternatively, handcrafted heuristics (e.g., the Fault-Tolerant Bottleneck Heuristic (FTBH) introduced in [101]) can be designed that aim to limit the complexity by restricting the search space at the expense of finding a suboptimal feasible solution. Metaheuristics (e.g., GRASP, see [46]) can be applied, too. In order to evaluate and compare the approaches, the resulting TSN element schedule configuration has to be evaluated according to the given objective function, since minimizing nondeterminism is the most important aspect of the scheduling problem (see Section 5). In the following, we analyze different reasons that lead to nondeterminism in order to establish a set of evaluation metrics for the resulting network-wide schedule.

6.1. Reasons for Nondeterminism in Time-Sensitive Networks

The main roots for nondeterminism in TSN are due to intrinsic network imprecision caused by, e.g., synchronization errors, imprecise information considered during scheduling (e.g., assuming incorrect bridge delays, etc.), and weak traffic temporal/spatial isolation. Another factor contributing to nondeterminism is the extrinsic inconsistency introduced by incoherent network control caused by, e.g., inconsistent network updates that could disrupt the smooth delivery of frames. We summarize the reasons for nondeterminism in Figure 9.

6.1.1. Synchronization Errors

TSN switches must be synchronized for coherent network schedules. Using gPTP, as defined in IEEE 802.1AS [127], TSN-enabled switches can achieve a very low synchronization error in the order of 10 ns. The gPTP messages exchanged between network entities are part of the network configuration traffic and, therefore, they are not considered the highest priority. Ref. [127] suggests the prioritization of synchronization messages but only stipulates a nonzero traffic class. Therefore, a nonoptimized GCL configuration in some of the switches could cause a large synchronization error or even the loss of synchronization due to timeouts. Synchronization errors can be problematic and may lead to frames missing the allocated cycle interval or the interleaving of frames, which could cause QoS violations.

6.1.2. Ineffective Temporal/Spatial Isolation

If the network is perfectly synchronized, nondeterminism could exist due to frame burst accumulation. In order to prevent this, interframe interaction must be controlled through frame prioritization, shaping, queuing, and gating. For LTS and BE traffic, spatial isolation (through prioritization/queuing and routing) and relaxed temporal isolation (through CBS shaping) is enough to fulfill stream timeliness requirements. In addition to spatial isolation, HTS traffic requires stricter temporal isolation through, e.g., gating or guard-banding. Those techniques, among others, provide the ability to control the timing but they need to be properly configured. Thus, a better schedule is a schedule that applies stricter temporal/spatial traffic isolation.

6.1.3. Inconsistent Network Configuration Updates

Some use cases may require a dynamic update of the TSN configuration during runtime (e.g., if a base-cycle with periodic updates for the GCL is used, as suggested by [97], or if some of the end-hosts are wireless and changing their location in the network [128]). This is a very complex task, as already scheduled streams shall not violate their constraints during such network updates. In addition, the new configuration must also respect the constraints in terms of timeliness. Finally, the update process should not result in packet loss during and after the update is activated. As every network update results in costs, such costs should be minimized. The configuration update could happen in the whole network at once but this needs a high degree of synchronization in order to decide when to start applying the new configuration. A simpler approach would be to perform the network update in several steps that need to be orchestrated. In the case that streams need to be rerouted during configuration updates, IEEE802.1CB [129] could be used to make that process seamless. In both cases, frequent updates of the GCL, as in [97], may lead to severe problems. Ref. [28] showed that configuration updates can not be lossless and duplicate-free at the same time and suggested a reconfiguration procedure to prevent either of them. However, independently of the approach, a configuration update should guarantee a smooth continuation of the transmission operations in the network while accounting for the queuing delay and avoiding interference.

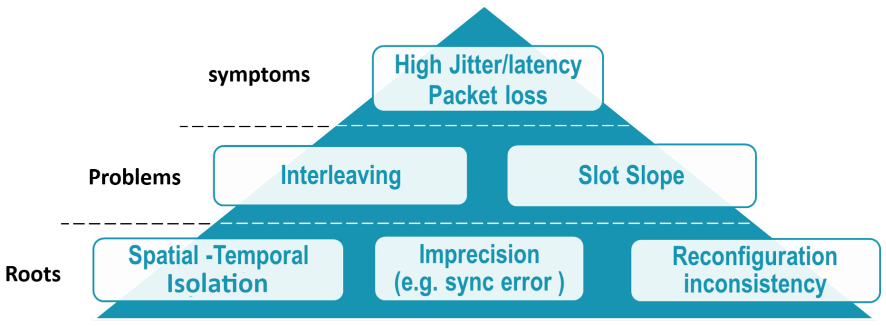

symptoms
High Jitter/latency
Packet loss
Problems
Interleaving
Slot Slope
Roots
Spatial -Temporal Isolation
Imprecision (e.g. sync error)
Reconfiguration inconsistency

Figure 9. Nondeterminism pyramid.

The abovementioned root causes of nondeterminism may lead to the following problems:

Slot slope problem: A slot slope happens when a frame is ready for transmission, whether before or after the opening of its respective transmission window, as shown in Figure 10. This slope could be caused by a synchronization error or precision error in the planned gate opening timestamp. Large slopes result in frames missing their allotted transmission windows. Small slopes could be mitigated by the scheduling algorithm by introducing a tolerance interval to the length of the transmission window. The amount of tolerance is a scheduling choice. Bigger tolerance intervals (i.e., larger HTS slots) reduce the effect of slope since they reserve more time for HTS transmission. The extra transmission time could be used by the other traffic types. Therefore, a large HTS slot reduces the chances of LTS and BE traffic. On the other hand, very small tolerance (i.e., small HTS slots) could cause the frames to miss their allotted times easily. The lookahead mechanism checks if the remaining time is enough to send the frame before the gate closes, otherwise it blocks the sending. A synchronization error or an incomplete previous transmission (especially if guard-banding is not applied) could trigger the lookahead mechanism to consider the small HTS slot insufficient and deny the transmission. Paths with many hops can contribute to slope accumulation as well, especially if the HTS slots are too wide. Thus, a schedule should have the appropriate amount of tolerance without exaggerations.

Interleaving problem: Interleaving of frames is explained in [38]. Nevertheless, we mention it briefly here, and for more detail, we refer the reader to [38]. Frame interleaving is an intratraffic-class problem. In the case that several HTS streams are using the same queue (no spatial isolation) and are having expected arrival times that are relatively close (weak temporal isolation), the order of the frames in the queue may be nondeterministic. This could result in different order from one cycle to another, which may lead to variable queuing delay and higher jitter. Frame interleaving in a switch port may lead to slot slope in the next switch. Thus, a schedule should aim to reduce the probability of frame interleaving.

In general, the reasons for nondeterminism are correlated and may exacerbate along the path. Figure 9 summarizes the root causes, problems, and manifestation of nondeterminism.

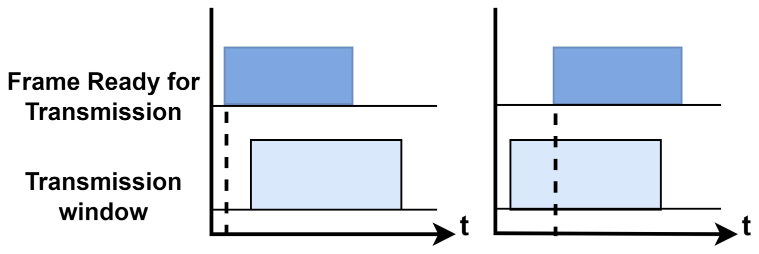


Figure 10. Examples of the slot slope problem.

6.2. Evaluation Metrics

Different aspects can be used in order to evaluate a TSN schedule that determines the network configuration. While we argue that determinism is the most important feature that is added by the TSN standards, more attention should be given to it when formulating the problem. For the evaluation of a schedule, determinism is also a very important criterion. However, most of the surveyed approaches do not really evaluate the quality of the resulting schedule. As shown in Table 5, the top two most occurring evaluation metrics are runtime and schedulability, while jitter is at number six in the ranking.

Table 5. Frequency of occurrence of metrics in the surveyed literature.

Metric	Occurrence in Literature
Runtime	70
Schedulability	45
Latency	40
Objective specific	33
Resource utilization (memory, CPU, link/bandwidth/GB, network)	19
Jitter	10
Scalability	7
Schedule length	5
Optimality	3

In this section, we suggest a set of evaluation metrics for TSN scheduling. As shown in Figure 11, we consider four different metric classes. The first and most important one concerns the evaluation of the schedule against the traffic requirements. This is because a schedule is considered feasible and can be applied only if it satisfies the QoS requirements of the streams. The second level of evaluation metrics concerns the quality of the schedule in relation to its determinism. The more deterministic the schedule is, the better the guarantees that the network can give to the users. The next level of evaluation treats the objective specific to the work. As different works target the optimization of different objectives, it is important to evaluate how optimal the solution is according to the chosen objective function. The three previous levels of evaluation consider the performance of the algorithm according to its output. The fourth level evaluates the algorithm independently of its output. In the following, we present a noncomprehensive list of metrics that could be used in each level of evaluation.

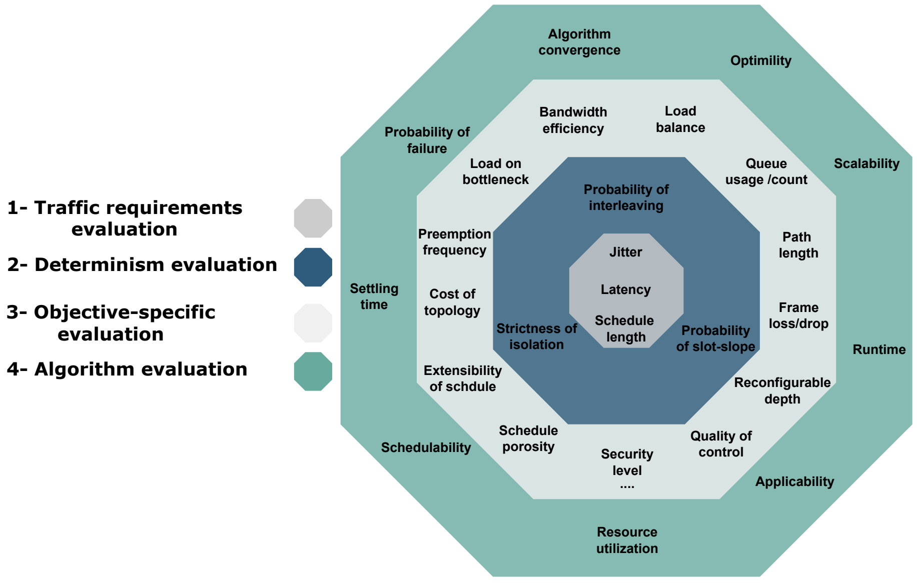


Figure 11. Classification of evaluation metrics.

The requirements of the use case could be expressed as constraints for the problem formulation. The following metrics could be used to evaluate them:

- Worst-case latency [11,15,24,29,33,35–37,40,46–49,53,61,62,70,72,73,75,76,80,81,83,86–88,91,97,99,100,104–106,110,111,116,118,119,123];
- Worst-case jitter [29,41,53,62,80,81,84,86,87,104];
- Maximum schedule length [37,42,73,81,90,97].

Afterward, the evaluation of the schedule should be against the optimization objectives. Metrics evaluating the primary objective, i.e., determinism, include:

- The strictness of temporal and spatial isolation;
- The probability of slot slope;
- The probability of frames/streams interleaving;
- Average jitter [41,62].

Metrics at the third level could be very different as they evaluate the secondary objectives that are specific to each work. In the following, we list some of the ones used in the literature as examples:

- Load balance (expressed in [33,59] as slot occupancy, and in [34,40,75,87,96] as link/network utilization);
- Bandwidth efficiency(expressed in [90] as bandwidth loss and in [80,91] as bandwidth utilization);
- Load on the bottleneck (expressed in [88] as the number of available flows);
- Frequency of preemption [95];
- Cost of topology [30,126];
- Extensibility of the schedule [12] (measures the deviation in the duration of unused spaces in the schedule);
- Porosity of the schedule [52] (measures the spacing between HTS windows to allow LTS and BE transmission);
- Security level [104];
- Quality of control (expressed in [85,107] as update duration and expressed in [86] as execution jitter of control application);
- Reconfigurable depth(expressed in [67] as the number of successful global reconfigurations because of core failure before the reconfiguration operation is unfeasible);
- Frame loss/drop [83,107];
- Path length [53];
- Queue usage/count [32,44,46,57,74,118].

The evaluation of the schedule quality is a very important step before applying the configuration of the TSN elements to the network. This operation could be performed in the Central Network Controller (CNC) (e.g., in the TSN microservice according to the architecture presented in [89]). However, those metrics are not criteria for comparison between different scheduling algorithms since the schedule evaluation is use case-dependent. Consequently, different algorithms could perform better with some specific use cases but not with others. The evaluation of the scheduling algorithm/problem formulation could be based on metrics including:

- Optimality gap of the solution in relation to the objective function [74,112];
- Scalability [15,29,37,41,50,116,124] (expressed in [116] as the number of generated constraints and in [37] as runtime with regards to the number of flows);
- Runtime for a given problem complexity determined by the size of the topology and the number of streams to schedule [11,14,15,24,29–33,35,37,38,40–42,44–51,53,54,57–60, 62,65–68,70–74,76–78,82,84,86,87,91,95–98,101,104–107,109,110,112,113,116–124,126];
- Scope of applicability of the algorithm based on its assumptions (e.g., algorithms with very restrictive assumptions are not applicable in the general case but for specific use cases);
- Resource utilization (e.g., CPU, memory, etc.) [61,109,119];
- The schedulability (ratio of scheduled streams per traffic type) under variable traffic loads [14,29,31,32,35,40,42,45–47,49,58,60,65–72,74,75,77,85,87,93,95,96,100–102,105–107,112,113,117, 118,120–124,126];
- Settling time [85];
- Probability of failure per hour [102];
- Algorithm convergence [82].

The evaluation model that we suggest in this section could be used as a reference to assess the depth of the evaluation. Although most of the works use metrics from the first and second levels, we suggest giving more importance to the determinism of communication in the evaluation and consider it the second most important after the satisfaction of the hard constraints.

7. Summary and Conclusions

This paper provides an extensive overview of the configuration of TSN networks, specifically the shaping part. For proper configuration of TSN shapers, the scheduling problem has to be formulated and solved. Different formulation approaches exist in the literature, which makes it difficult to compare the problem resolution algorithms. In this work, we aimed to classify the different approaches and evaluate the different problem formulations. To this extent, we first presented the possible configurations, the traffic categories, and the shaping options. Then, we presented the challenges and the utility of every shaping choice and we suggested a reasonable mapping between traffic categories and shapers. We focused on scheduling of Time-Aware Shaping, which requires solving a network-wide optimization problem. To help build a clear understating of the scheduling task in TSN networks, we surveyed more than 100 solution approaches in order to analyze the evolution of the state of the art. We deeply analyzed and classified the proposed solutions and presented our perspective on future research directions and the challenges related to TSN scheduling. Finally, we conducted a literature-based analysis of the objectives, constraints, and assumptions of the TAS scheduling problem, we analyzed the nondeterminism in the TSN network, and we presented a possible evaluation plan with a structured set of metrics.

In future work, we aim for a mathematical formulation of the suggested evaluation metrics (i.e., the interleaving problem probability, the slot slope probability, etc.) and a comparison of different algorithms under the same setup. Another important direction that we are focusing on is the study of the control plane design and management operations. Here, the performance of the control plane is mainly determined by the complexity of the configuration synthesizing process which requires solving ofr the network-wide scheduling problem and the time to push the configuration to the TSN elements using protocols such as Netconf/Yang. Finally, reconfiguring TSN elements during runtime is another important aspect that we aim to tackle.

Author Contributions: Writing—original draft, H.C.; Writing—review & editing, A.K.; Supervision, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: Parts of this work have been supported by the Knowledge Foundation of Sweden (KKS) through the Synergi Project AIDA. Additional funding has been provided by the Bavarian State Ministry of Education and Culture, Science and Art through the High-Tech Agenda (HTA).

Data Availability Statement: The data presented in this study is based on the published papers referenced in the bibliography section.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Summary of the Surveyed Scheduling Works

Table A1. An overview on scheduling solutions—part 1.

Year	Paper	Tasks	Approach	Mode	Model	Shaper
2016	[38]	S	SMT/OMT, Z3	off	C	TAS
	[37]	S	ILP, job-shop prob.	off	C	TAS
	[50]	S R	Dijkstra and ILP for HTS, GRASP for AVB routing	off	C	TAS
	[34]	S	UBS traffic-urgency-based scheduler	off	D	ATS
	[78]	S A	First-order logic, SMT, MIP, incremental scheduling	off	C	TAS
2017	[58]	S R	ILP, SMT/OMT, UBS, base-period, full-network traversal	off/on	C	TAS
	[79]	n.s	SMT-based	off	C	TAS
	[39]	S	Graphical modeling, model-based inference of rules and constraints, logic programming, SMT	off	C	TAS
	[92]	n.s	SMT/OMT	off	C	TAS
	[51]	S R	Multiobjective optimization, 0-1 ILP, genetic algorithm	off	C	TAS
	[40]	S R	ILP formulation	off	C	TAS
	[44]	S	Extracts flow properties, react to traffic changes, heuristic	off	C	TAS
	[30]	R T	GRASP metaheuristic, constraint programming, topology and routing heuristic	off	C	n.s
2018	[14]	S R	ILP, dummy objective variable	off	C	TAS
	[15]	R	Study the effect of routing on schedulability, ILP-based	off	C	TAS
	[41]	S	First-order theory of arrays, window-based, SMT/OMT	off	C	TAS
	[45]	S	GRASP metaheuristic	off	C	TAS
	[46]	S R	K-shortest path (KSP), GRASP metaheuristic	off	C	TAS
	[29]	S R	SMT, route subsets and time slices heuristic	off	C	TAS
	[47]	S R A	Genetic algorithm, task bindings	off	C	TAS
	[57]	S	Configuration agent, list-based heuristic for fog computing	on	C	TAS
	[126]	S R T	Greedy heuristic, iterative path selection, Yen's algorithm	off	C	TAS

Table A1. *Cont.*

Year	Paper	Tasks	Approach	Mode	Model	Shaper
2019	[62]	S	Flow fragmentation, SMT Z3	off	C	TAS
	[48]	S	Problem segmentation, relaxed constraints, SMT, incremental approach	off	C	TAS
	[42]	S	Queue assignment, SMT/OMT, fragmentation, heuristics: no-gate-closing, strict-priority, move-forward,	off	C	TAS
	[49]	S R	Heuristic list scheduler	off	C	TAS
	[103]	S	ILP, time redundancy	off	C	TAS
	[124]	S R	Incremental algorithms, QoS-aware path selection, SWOTS-AEAP, SWOTS-ASAP, SWTS, SWOTS-WS	on	C	TAS
	[104]	S R	Binary search, genetic algorithm, incremental synthesis, security optimization	off	C	TAS
2020	[97]	S R	Base-period, periodic update of the GCL, SMT	on	C	TAS
	[59]	S R	Load balancing, MIP, vector bin packing (VBP)	off	C	TAS
	[11]	S	Injection time grouping algorithm, parameter selection	off	C	TAS,CQF
	[123]	S	Iterative, windows-based, no traffic isolation	off	C	TAS
	[117]	S R	Incremental scheduling, streams partitioning, iterated ILP, degree of conflict awareness	off	C	TAS
2020	[96]	S R	MDPC, MDTC, heuristics (HB-S, HD-S), BFS	off/on	C	TAS
	[32]	S	Injection time planning mechanism, Tabu-ITP	off	C	CQF
	[61]	S	Extensibility-aware scheduling (EASA) algorithm, simulated annealing-based metaheuristic	off	C	TAS
	[36]	S	Suggest enhancement to TAS: protection band	off	C	TAS
	[118]	S	Frame check at egress queue, hardware enhancement	off	C	TAS
	[111]	S	SMT	off	C	TAS
	[109]	S R A	Constraint-programming-based, TESLA protocol	off	C	TAS
	[84]	S	Theory of computation: constraint and logic programming, systematic, and metaheuristic search	off	C	TAS
	[63]	S	Constraint programming, joint TSN and 5G scheduling	off	D	TAS
	[122]	S R	SMT/OMT, co-design approach (CSRST), SHLS	off	C	TAS
	[76]	S R	Hybrid genetic algorithm, compress schedules	off	C	TAS
	[98]	S R	Conflict-graph-based approach	off	C	TAS
	[112]	S R	Optimality gap, shortest path, ILP	off	C	TAS
	[69]	S R	AGRS approach, simplify topology	off/on	C	TAS
	[85]	S	Fixed-priority scheduling (FPS), co-design optimization	off	C	TAS

Table A2. Overview on scheduling solutions—part 2.

Year	Paper	Tasks	Approach	Mode	Model	Shaper
2021	[60]	S R	Vector bin packing(VBP), bottleneck heuristic, MML heuristic, coefficient of variation heuristic, MDP heuristic	on	C	TAS
	[87]	S R	Concurrent multipath transmission, MMF heuristic	off	C	TAS
	[24]	S	Z3 SMT/OMT, OMNeT++ simulation, increased porosity	off	C	TAS
	[53]	S R	Ant colony optimization, binary multi-knapsack formulation, improved genetic simulated	off	C	TAS
	[54]	S A	SMT, application level optimization, tasks transmission	off	C	TAS
	[99]	n.s.	Light traffic heuristic, systematic procedure	off	C	TAS

Table A2. *Cont.*

Year	Paper	Tasks	Approach	Mode	Model	Shaper
	[101]	S R	FTB heuristic, vector bin packing (VBP), frame replication	on	C	TAS
	[35]	S	Physical queues abstraction, two stages stream-level scheme	off	C	TAS
	[66]	S R	Incremental algorithm (IRAS), prerouting algorithm	on	C	TAS
	[86]	S	CP, Tabu Search, diversification and intensification	off	C	TAS
	[65]	S R	Dynamic programming, ultra-fast adaptive greedy method	off/on	C	CQF
	[75]	R	Real-time routing scheduler (RTRS)	off	C	TAS
	[74]	S R	Constraint programming, logic-based decomposition	off	C	TAS
	[80]	S	Genetic algorithm	off	C	TAS
	[105]	S	Z3 SMT/OMT, design space exploration problem	off	C	TAS
	[102]	S	Redundant transmission, PFT-TSN, SMT	off	C	TAS
	[107]	S	ILP, enhanced online/offline schedulers	off/on	C	TAS
	[90]	S	Constraint strategy	off	C	TAS
	[77]	S	Low-delay message fragmentation, no-wait scheduling, OMT	off	C	TAS
	[93]	S	Semisupervised-learning, iterative, stream partitioning, sparse encoding	off	C	TAS
	[106]	S R	SMT, path redundancy, incremental synthesis heuristic	off	C	TAS
	[110]	S R	SMT, ASIL decomposition, binary search, incremental	off	C	TAS
	[88]	S R	Heuristics: greedy multicast flows routing, backtracking, incremental, as-early-as-possible, iterative local search, redundancy-based	off	C	TAS
	[70]	S R	Cluster-ILP, incremental scheduling, divide and conquer	off/on	C	TAS
2022	[33]	S	Flow sequences analysis, parallel, incremental computing, FLJ-VB, divisibility theory, PD-based search boundary	off	C	CQF
	[94]	S	Cycle model normalization, cooperative resource allocation	off	C	C(S)QF
	[83]	S	Machine learning , regression-based surrogate model, NSGA-II, SMPSO, multiobjective optimizations	off	C	TAS, CBS
	[71]	S	Partial schedules, conflicts-guided space probing, EPIC	off	C	TAS
	[91]	S	CP, worst-case delay analysis window-based	off	C	TAS
	[81]	S	Genetic algorithm	off	C	TAS
	[82]	S	Deep reinforcement learning, no-wait scheduling, A3C	off	C	TAS
	[116]	S	CP, Exclusive queue allocation, size-based isolation	off	C	TAS
	[68]	S	HERMES heuristic, zero or relaxed reception jitter	on	C	TAS
	[119]	S R A	TESLA heuristic, redundant disjunct paths, CP, Simulated Annealing, ASAP list scheduling	off	C	TAS
	[120]	S R	Dijkstra, SMT, single-fault, temporal exploration	off	C	TAS
	[72]	S R	ILP, cycle correlation, classification, adaptive cycle	off	C	TAS
	[121]	S R	LSSR-architecture, stream partition, graph-clustering	off	C	TAS
	[95]	S R	SMT-based, combine TAS with preemption	off	C	TAS
	[73]	S R	Routing to avoid congestion and reduce e2e latency	off	C	TAS
	[67]	S R A	Joint-ILP, SCA heuristic, reconfigurable depth metric	on	C	TAS
	[100]	S R	Reinforcement learning, JRSA heuristic	off	C	CQF

References

1. PROFINET Technology and Application–System Description. Available online: https://www.profibus.com/download/profinet-technology-and-application-system-description (accessed on 1 October 2023).
2. Seol, Y.; Hyeon, D.; Min, J.; Kim, M.; Paek, J. Timely Survey of Time-Sensitive Networking: Past and Future Directions. *IEEE Access.* **2021**, *9*, 142506–142527. [CrossRef]

3. Ashjaei, M.; Bello, L.L.; Daneshtalab, M.; Patti, G.; Saponara, S.; Mubeen, S. Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities. *J. Syst. Archit.* **2021**, *117*, 102137. [CrossRef]
4. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; El Bakoury, H. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research. *IEEE Commun. Surv. Tutorials* **2018**, *21*, 88–145. [CrossRef]
5. Bello, L.L.; Steiner, W. A perspective on IEEE time-sensitive networking for industrial communication and automation systems. *Proc. IEEE* **2019**, *107*, 1094–1120. [CrossRef]
6. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M. Performance Comparison of IEEE 802.1 TSN Time Aware Shaper (TAS) and Asynchronous Traffic Shaper (ATS). *IEEE Access* **2019**, *7*, 44165–44181. [CrossRef]
7. Thangamuthu, S.; Concer, N.; Cuijpers, P.J.; Lukkien, J.J. Analysis of ethernet-switch traffic shapers for in-vehicle networking applications. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 9–13 March 2015; pp. 55–60.
8. Steiner, W.; Craciunas, S.S.; Oliver, R.S. Traffic planning for time-sensitive communication. *IEEE Commun. Stand. Mag.* **2018**, *2*, 42–47. [CrossRef]
9. Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. Rfc2475: An Architecture for Differentiated Service. Available online: https://datatracker.ietf.org/doc/html/rfc2475 (accessed on 13 January 2023).
10. IEC/IEEE 60802 TSN Profile for Industrial Automation, D1.3 draft. Available online: https://1.ieee802.org/tsn/iec-ieee-60802 (accessed on 7 August 2021).
11. Zhang, J.; Xu, Q.; Lu, X.; Zhang, Y.; Chen, C. Coordinated data transmission in Time-Sensitive Networking for Mixed Time-Sensitive Applications. In Proceedings of the IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, Singapore, 18 October 2020; pp. 3805–3810.
12. *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*; IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption; IEEE: Piscataway, NJ, USA, 2016; pp. 1–52. [CrossRef]
13. *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*; IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic; IEEE: Piscataway, NJ, USA, 2016; pp. 1–57. [CrossRef]
14. Falk, J.; Dürr, F.; Rothermel, K. Exploring practical limitations of joint routing and scheduling for TSN with ILP. In Proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hakodate, Japan, 28–31 August 2018; pp. 136–146.
15. Nayak, N.G.; Duerr, F.; Rothermel, K. Routing algorithms for IEEE802. 1Qbv networks. *ACM SIGBED Rev.* **2018**, *15*, 13–18. [CrossRef]
16. Zhao, L.; Pop, P.; Craciunas, S.S. Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus. *IEEE Access* **2018**, *6*, 41803–41815. [CrossRef]
17. Zhao, L.; Pop, P.; Gong, Z.; Fang, B. Improving Latency Analysis for Flexible Window-Based GCL Scheduling in TSN Networks by Integration of Consecutive Nodes Offsets. *IEEE Internet Things J.* **2020**, *8*, 5574–5584. [CrossRef]
18. *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*; IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks; IEEE: Piscataway, NJ, USA, 2018; pp. 1–1993. [CrossRef]
19. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*; IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams; IEEE: Piscataway, NJ, USA, 2010; pp. C1–C72. [CrossRef]
20. *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*; IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping; IEEE: Piscataway, NJ, USA, 2020; pp. 1–151. [CrossRef]
21. *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*; IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding; IEEE: Piscataway, NJ, USA, 2017; pp.1–30. [CrossRef]
22. Kim, M.; Hyeon, D.; Paek, J. eTAS: Enhanced Time-Aware Shaper for Supporting Non-Isochronous Emergency Traffic in Time-Sensitive Networks. *IEEE Internet Things J.* **2021**, *9*, 21860262. [CrossRef]
23. Tindell, K.W.; Burns, A.; Wellings, A.J. Allocating hard real-time tasks: An NP-hard problem made easy. *Real-Time Syst.* **1992**, *4*, 145–165. [CrossRef]
24. Houtan, B.; Ashjaei, M.; Daneshtalab, M.; Sjödin, M.; Mubeen, S. Synthesising schedules to improve QoS of best-effort traffic in TSN networks. In Proceedings of the 29th International Conference on Real-Time Networks and Systems, Virtual, 7–9 April 2021; pp. 68–77.
25. Houtan, B.; Ashjaei, M.; Daneshtalab, M.; Sjödin, M.; Afshar, S.; Mubeen, S. Schedulability Analysis of Best-Effort Traffic in TSN Networks. In Proceedings of the 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, 7–10 September 2021; pp. 1–8.
26. Zhao, L.; Pop, P.; Steinhorst, S. Quantitative performance comparison of various traffic shapers in time-sensitive networking. *arXiv* **2021**, arXiv:2103.13424.

27. Pruski, A.; Ojewale, M.A.; Gavrilut, V.; Yomsi, P.M.; Berger, M.S.; Almeida, L. Implementation Cost Comparison of TSN Traffic Control Mechanisms. In Proceedings of the 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, 7–10 September 2021; pp. 1–8.
28. Kohler, T.; Dürr, F.; Rothermel, K. Consistent network management for software-defined networking based multicast. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 447–461. [CrossRef]
29. Mahfouzi, R.; Aminifar, A.; Samii, S.; Rezine, A.; Eles, P.; Peng, Z. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition, Dresden, Germany, 19–23 March 2018; pp. 682–687.
30. Gavrilut, V.; Zarrin, B.; Pop, P.; Samii, S. Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking. In Proceedings of the 25th International Conference on Real-Time Networks and Systems 2017, Grenoble, France, 4–6 October 2017; pp. 267–276.
31. Lin, Y.; Jin, X.; Zhang, T.; Han, M.; Guan, N.; Deng, Q. Queue assignment for fixed-priority real-time flows in time-sensitive networks: Hardness and algorithm. *J. Syst. Archit.* **2021**, *116*, 102141. [CrossRef]
32. Yan, J.; Quan, W.; Jiang, X.; Sun, Z. Injection time planning: Making cqf practical in time-sensitive networking. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 616–625.
33. Zhang, Y.; Xu, Q.; Xu, L.; Chen, C.; Guan, X. Efficient Flow Scheduling for Industrial Time-Sensitive Networking: A Divisibility Theory Based Method. *IEEE Trans. Ind. Inform.* **2022**, *18*, 22136129. [CrossRef]
34. Specht, J.; Samii, S. Urgency-based scheduler for time-sensitive switched ethernet networks. In Proceedings of the 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), Toulouse, France, 5–8 July 2016; pp. 75–85.
35. Xue, J.; Shou, G.; Liu, Y.; Hu, Y.; Guo, Z. Time-Aware Traffic Scheduling with Virtual Queues in Time-Sensitive Networking. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 604–607.
36. Kim, M.; Min, J.; Hyeon, D.; Paek, J. TAS scheduling for real-time forwarding of emergency event traffic in TSN. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 21–23 October 2020; pp. 1111–1113.
37. Dürr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212.
38. Craciunas, S.S.; Oliver, R.S.; Chmelík, M.; Steiner, W. Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks. InProceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 183–192.
39. Farzaneh, M.H.; Kugele, S.; Knoll, A. A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017; pp. 1–8.
40. Schweissguth, E.; Danielis, P.; Timmermann, D.; Parzyjegla, H.; Mühl, G. ILP-based joint routing and scheduling for time-triggered networks. In Proceedings of the 25th International Conference on Real-Time Networks and Systems 2017, Grenoble, France, 4–6 October 2017; pp. 8–17.
41. Oliver, R.S.; Craciunas, S.S.; Steiner, W. IEEE 802.1 Qbv gate control list synthesis using array theory encoding. In Proceedings of the 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, Portugal, 11–13 April 2018; pp. 13–24.
42. Jin, X.; Xia, C.; Guan, N.; Xu, C.; Li, D.; Yin, Y.; Zeng, P. Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries. *IEEE Access* **2020**, *8*, 6751–6767. [CrossRef]
43. Hellmanns, D.; Haug, L.; Hildebr, M.; Dürr, F.; Kehrer, S.; Hummen, R. How to optimize joint routing and scheduling models for TSN using integer linear programming. In Proceedings of the 29th International Conference on Real-Time Networks and Systems 2021, Virtual, 7–9 April 2021; pp. 100–111.
44. Raagaard, M.L.; Pop, P.; Gutiérrez, M.; Steiner, W. Runtime reconfiguration of time-sensitive networking (TSN) schedules for fog computing. In Proceedings of the 2017 IEEE Fog World Congress (FWC), Santa Clara, CA, USA, 30 October–1 November 2017; pp. 1–6.
45. Gavriluţ, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 13–15 June 2018; pp. 1–4.
46. Gavriluţ, V.; Zhao, L.; Raagaard, M.L.; Pop, P. AVB-aware routing and scheduling of time-triggered traffic for TSN. *IEEE Access* **2018**, *6*, 75229–75243. [CrossRef]
47. Pahlevan, M.; Obermaisser, R. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, Italy, 4–7 September 2018; pp. 337–344.
48. Pozo, F.; Rodriguez-Navas, G.; Hansson, H. Methods for large-scale time-triggered network scheduling. *Electronics* **2019**, *8*, 738. [CrossRef]
49. Pahlevan, M.; Tabassam, N.; Obermaisser, R. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM Sigbed Rev.* **2019**, *16*, 15–20. [CrossRef]
50. Pop, P.; Raagaard, M.L.; Craciunas, S.S.; Steiner, W. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Phys. Syst. Theory Appl.* **2016**, *1*, 86–94. [CrossRef]

51. Smirnov, F.; Glaß, M.; Reimann, F.; Teich, J. Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks. In Proceedings of the 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 18–22 June 2017; pp. 1–6.
52. Steiner, W. Synthesis of static communication schedules for mixed-criticality systems. In Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, Newport Beach, CA, USA, 28–31 March 2011; pp. 11–18.
53. Wang, Y.; Chen, J.; Ning, W.; Yu, H.; Lin, S.; Wang, Z.; Pang, G.; Chen, C. A time-sensitive network scheduling algorithm based on improved ant colony optimization. *Alex. Eng. J.* **2021**, *60*, 107–114. [CrossRef]
54. Feng, T.; Yang, H. SMT-based Task-and Network-level Static Schedule for Time Sensitive Network. In Proceedings of the 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), Beijing, China, 14–16 May 2021; pp. 764–770.
55. Craciunas, S.S.; Oliver, R.S. SMT-based task-and network-level static schedule generation for time-triggered networked systems. In Proceedings of the 22nd International Conference on Real-Time Networks and Systems 2014, Versailles, France, 8–10 October 2014; pp. 45–54.
56. Craciunas, S.S.; Oliver, R.S.; Ecker, V. Optimal static scheduling of real-time tasks on distributed time-triggered networked systems. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA) 2014, Barcelona, Spain, 16–19 September 2014; pp. 1–8.
57. Pop, P.; Raagaard, M.L.; Gutierrez, M.; Steiner, W. Enabling fog computing for industrial automation through time-sensitive networking (TSN). *IEEE Commun. Stand. Mag.* **2018**, *2*, 55–61. [CrossRef]
58. Nayak, N.G.; Dürr, F.; Rothermel, K. Incremental flow scheduling and routing in time-sensitive software-defined networks. *IEEE Trans. Ind. Inform.* **2017**, *14*, 2066–2075. [CrossRef]
59. Syed, A.A.; Ayaz, S.; Leinmüller, T.; Chra, M. MIP-based Joint Scheduling and Routing with Load Balancing for TSN based In-vehicle Networks. In Proceedings of the 2020 IEEE Vehicular Networking Conference (VNC), New York, NY, USA, 16–18 December 2020; pp. 1–7.
60. Syed, A.A.; Ayaz, S.; Leinmüller, T.; Chandra, M. Dynamic Scheduling and Routing for TSN based In-vehicle Networks. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
61. Barzegaran, M.; Karagiannis, V.; Avasalcai, C.; Pop, P.; Schulte, S.; Dustdar, S. Towards extensibility-aware scheduling of industrial applications on fog nodes. In Proceedings of the 4th IEEE International Conference on Edge Computing, Melbourne, VIC, Australia, 11–14 May 2020; pp. 1–9.
62. Dos Santos, A.C.; Schneider, B.; Nigam, V. TSNSCHED: Automated schedule generation for time sensitive networking. In Proceedings of the 2019 Formal Methods in Computer Aided Design (FMCAD), San Jose, CA, USA, 22–25 October 2019; pp. 69–77.
63. Ginthör, D.; Guillaume, R.; von Hoyningen-Huene, J.; Schüngel, M.; Schotten, H.D. End-to-end optimized joint scheduling of converged wireless and wired time-sensitive networks. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 222–229.
64. Pozo, F.; Rodriguez-Navas, G.; Hansson, H. Schedule reparability: Enhancing time-triggered network recovery upon link failures. In Proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hakodate, Japan, 28–31 August 2018; pp. 147–156.
65. Krolikowski, J.; Martin, S.; Medagliani, P.; Leguay, J.; Chen, S.; Chang, X.; Geng, X. Joint routing and scheduling for large-scale deterministic IP networks. *Comput. Commun.* **2021**, *165*, 33–42. [CrossRef]
66. Huang, Y.; Wang, S.; Huang, T.; Wu, B.; Wu, Y.; Liu, Y. Online Routing and Scheduling for Time-Sensitive Networks. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 7–10 July 2021; pp. 272–281.
67. Li, J.; Xiong, H.; Li, Q.; Xiong, F.; Feng, J. Run-Time Reconfiguration Strategy and Implementation of Time-Triggered Networks. *Electronics* **2022**, *11*, 1477. [CrossRef]
68. Bujosa, D.; Ashjaei, M.; Papadopoulos, A.V.; Nolte, T.; Proenza, J. HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities. In Proceedings of the 30th International Conference on Real-Time Networks and Systems, Paris, France, 7–8 June 2022; pp. 70–80.
69. Yu, Q.; Gu, M. Adaptive group routing and scheduling in multicast time-sensitive networks. *IEEE Access* **2020**, *8*, 37855–37865. [CrossRef]
70. Li, C.; Zhang, C.; Zheng, W.; Wen, X.; Lu, Z.; Zhao, J. Joint Routing and Scheduling for Dynamic Applications in Multicast Time-Sensitive Networks. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
71. Vlk, M.; Brejchová, K.; Hanzálek, Z.; Tang, S. Large-scale periodic scheduling in time-sensitive networks. *Comput. Oper. Res.* **2022**, *137*, 105512. [CrossRef]
72. Nie, H.; Li, S.; Liu, Y. An Enhanced Routing and Scheduling Mechanism for Time-Triggered Traffic with Large Period Differences in Time-Sensitive Networking. *Appl. Sci.* **2022**, *12*, 4448. [CrossRef]
73. Li, Y.; Jiang, J.; Hong, S.H. Joint Traffic Routing and Scheduling Algorithm Eliminating the Nondeterministic Interruption for TSN Networks Used in IIoT. *IEEE Internet Things J.* **2022**, *9*, 18663–18680. [CrossRef]

74. Vlk, M.; Hanzálek, Z.; Tang, S. Constraint programming approaches to joint routing and scheduling in time-sensitive networks. *Comput. Ind. Eng.* **2021**, *157*, 107317. [CrossRef]
75. Chang, S.H.; Chen, H.; Cheng, B.C. Time-predictable routing algorithm for Time-Sensitive Networking: Schedulable guarantee of Time-Triggered streams. *Comput. Commun.* **2021**, *172*, 183–195. [CrossRef]
76. Arestova, A.; Hielscher, K.S.; German, R. Design of a hybrid genetic algorithm for time-sensitive networking. In Proceedings of the International Conference on Measurement, Modelling and Evaluation of Computing Systems 2020, Saarbrucken, Germany, 16–18 March 2020; pp. 99–117.
77. Jin, X.; Xia, C.; Guan, N.; Zeng, P. Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 478–490. [CrossRef]
78. Craciunas, S.S.; Oliver, R.S. Combined task-and network-level scheduling for distributed time-triggered systems. *Real-Time Syst.* **2016**, *52*, 161–200. [CrossRef]
79. Craciunas, S.S.; Oliver, R.S.; Steiner, W. Formal scheduling constraints for time-sensitive networks. *arXiv* **2017**, arXiv:1712.02246.
80. Kim, H.J.; Lee, K.C.; Lee, S. A Genetic Algorithm based Scheduling Method for Automotive Ethernet. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 13–16 October 2021; pp. 1–5.
81. Kim, H.J.; Lee, K.C.; Kim, M.H.; Lee, S. Optimal Scheduling of Time-Sensitive Networks for Automotive Ethernet Based on Genetic Algorithm. *Electronics* **2022**, *11*, 926. [CrossRef]
82. Wang, X.; Yao, H.; Mai, T.; Nie, T.; Zhu, L.; Liu, Y. Deep reinforcement learning aided no-wait flow scheduling in time-sensitive networks. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 10–13 April 2022; pp. 812–817
83. Bezerra, D.; de Oliveira Filho, A.T.; Rodrigues, I.R.; Dantas, M.; Barbosa, G.; Souza, R.; Kelner, J.; Sadok, D. A machine learning-based optimization for end-to-end latency in TSN networks. *Comput. Commun.* **2022**, *195*, 424–440. [CrossRef]
84. Barzegaran, M.; Zarrin, B.; Pop, P. Quality-of-control-aware scheduling of communication in TSN-based fog computing platforms using constraint programming. In Proceedings of the 2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020), Sydney, Australia, 21 April 2020.
85. Dai, X.; Zhao, S.; Jiang, Y.; Jiao, X.; Hu, X.S.; Chang, W. Fixed-priority scheduling and controller co-design for time-sensitive networks. In Proceedings of the 39th International Conference on Computer-Aided Design 2020, Virtual, 2–5 November 2020; pp. 1–9.
86. Barzegaran, M.; Pop, P. Communication scheduling for control performance in TSN-based fog computing platforms. *IEEE Access* **2021**, *9*, 50782–50797. [CrossRef]
87. Zheng, Y.; Wang, S.; Yin, S.; Wu, B.; Liu, Y. Mix-Flow Scheduling for Concurrent Multipath Transmission in Time-Sensitive Networking. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
88. Li, H.; Cheng, H.; Yang, L. Reliable Routing and Scheduling in Time-Sensitive Networks. In Proceedings of the 2021 17th International Conference on Mobility, Sensing and Networking (MSN), Exeter, UK, 13–15 December 2021; pp. 806–811.
89. Chahed, H.; Kassler, A.J. Software-Defined time-sensitive Networks Configuration and Management. In Proceedings of the 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Heraklion, Greece, 9–11 November 2021; pp. 124–128.
90. Dai, J.; Wang, Z.; Zhong, L. Research on Gating Scheduling of Time Sensitive Network Based on Constraint Strategy. *J. Phys. Conf. Ser.* **2021**, *1920*, 012089. [CrossRef]
91. Barzegaran, M.; Reusch, N.; Zhao, L.; Craciunas, S.S.; Pop, P. Real-Time Traffic Guarantees in Heterogeneous Time-sensitive Networks. In Proceedings of the 30th International Conference on Real-Time Networks and Systems 2022, Paris, France, 7–8 June 2022; pp. 46–57.
92. Craciunas, S.S.; Oliver, R.S.; Ag, T. An overview of scheduling mechanisms for time-sensitive networks. *Proc. Real-Time Summer Sch. ETR* **2017**, *2017*, 1551–3203.
93. Tu, J.; Xu, Q.; Xu, L.; Chen, C. SSL-SP: A Semi-Supervised-Learning-Based Stream Partitioning Method for Scale Iterated Scheduling in Time-Sensitive Networks. In Proceedings of the 2021 22nd IEEE International Conference on Industrial Technology (ICIT), Valencia, Spain, 10–12 March 2021; pp. 1182–1187.
94. Huang, Y.; Wang, S.; Huang, T.; Liu, Y. Cycle-based time-sensitive and deterministic networks: Architecture, challenges, and open issues. *IEEE Commun. Mag.* **2022**, *60*, 81–87. [CrossRef]
95. Zhou, Y.; Samii, S.; Eles, P.; Peng, Z. Time-Triggered Scheduling for Time-Sensitive Networking with Preemption. In Proceedings of the 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 17–20 January 2022; pp. 262–267.
96. Yu, Q.; Wan, H.; Zhao, X.; Gao, Y.; Gu, M. Online scheduling for dynamic VM migration in multicast time-sensitive networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3778–3788. [CrossRef]
97. Li, Q.; Li, D.; Jin, X.; Wang, Q.; Zeng, P. A simple and efficient time-sensitive networking traffic scheduling method for industrial scenarios. *Electronics* **2020**, *9*, 2131. [CrossRef]
98. Falk, J.; Dürr, F.; Rothermel, K. Time-triggered traffic planning for data networks with conflict graphs. In Proceedings of the 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Sydney, Australia, 21–24 April 2020; pp. 124–136.
99. Kim, H.J.; Choi, M.H.; Kim, M.H.; Lee, S. Development of an Ethernet-Based Heuristic Time-Sensitive Networking Scheduling Algorithm for Real-Time In-Vehicle Data Transmission. *Electronics* **2021**, *10*, 157. [CrossRef]

100. Liu, Y.; Cheng, Z.; Ren, J.; Yang, D. Joint Routing and Scheduling for CQF. In Proceedings of the 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 22–25 April 2022; pp. 1–5. [CrossRef]
101. Syed, A.A.; Ayaz, S.; Leinmüller, T.; Chra, M. Fault-Tolerant Dynamic Scheduling and Routing for TSN based In-vehicle Networks. In Proceedings of the 2021 IEEE Vehicular Networking Conference (VNC), Ulm, Germany, 10–12 November 2021; pp. 72–75.
102. Feng, Z.; Cai, M.; Deng, Q. An efficient pro-active fault-tolerance scheduling of ieee 802.1 qbv time-sensitive network. *IEEE Internet Things J.* **2021**, *9*, 14501–14510. [CrossRef]
103. Dobrin, R.; Desai, N.; Punnekkat, S. On fault-tolerant scheduling of time sensitive networks. In Proceedings of the 4th International Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS 2019), Stuttgart, Germany, 9 July 2019.
104. Mahfouzi, R.; Aminifar, A.; Samii, S.; Eles, P.; Peng, Z. Security-aware routing and scheduling for control applications on Ethernet TSN networks. *ACM Trans. Des. Autom. Electron. Systems (TODAES)* **2019**, *25*, 1–26. [CrossRef]
105. Craciunas, S.S.; Oliver, R.S. Out-of-sync Schedule Robustness for Time-sensitive Networks. In Proceedings of the 2021 17th IEEE International Conference on Factory Communication Systems (WFCS), Linz, Austria, 9–11 June 2021; pp. 75–82.
106. Zhou, Y.; Samii, S.; Eles, P.; Peng, Z. Reliability-aware scheduling and routing for messages in time-sensitive networking. *ACM Trans. Embed. Comput. Syst. TECS* **2021**, *20*, 1–24. [CrossRef]
107. Pang, Z.; Huang, X.; Li, Z.; Zhang, S.; Xu, Y.; Wan, H.; Zhao, X. Flow scheduling for conflict-free network updates in time-sensitive software-defined networks. *IEEE Trans. Ind. Inform.* **2020**, *17*, 1668–1678. [CrossRef]
108. Zhang, Y.; Wu, J.; Liu, M.; Tan, A. TSN-based routing and scheduling scheme for Industrial Internet of Things in underground mining. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105314. [CrossRef]
109. Reusch, N.; Pop, P.; Craciunas, S.S. Work-in-progress: Safe and secure configuration synthesis for TSN using constraint programming. In Proceedings of the 2020 IEEE Real-Time Systems Symposium (RTSS), Houston, TX, USA, 1–4 December 2020; pp. 387–390.
110. Zhou, Y.; Samii, S.; Eles, P.; Peng, Z. ASIL-decomposition based routing and scheduling in safety-critical time-sensitive networking. In Proceedings of the 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS), Nashville, TN, USA, 18–21 May 2021; pp. 184–195.
111. Zhou, W.; Li, Z. Implementation and Evaluation of SMT-based Real-time Communication Scheduling for IEEE 802.1 Qbv in Next-generation in-vehicle network. In Proceedings of the 2020 2nd International Conference on Information Technology and Computer Application (ITCA), Guangzhou, China, 18–20 December 2020; pp. 457–461.
112. Schweissguth, E.; Timmermann, D.; Parzyjegla, H.; Danielis, P.; Mühl, G. ILP-based routing and scheduling of multicast realtime traffic in time-sensitive networks. In Proceedings of the 2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Gangnueng, Republic of Korea, 19–21 August 2020; pp. 1–11.
113. Berisa, A.; Zhao, L.; Craciunas, S.S.; Ashjaei, M.; Mubeen, S.; Daneshtalab, M.; Sjödin, M. AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption. In Proceedings of the 30th International Conference on Real-Time Networks and Systems 2022, Paris, France, 7–8 June 2022; pp. 207–218.
114. 5G Alliance for Connected Industries and Automation (5g-ACIA). Integration of 5G with Time-Sensitive Networking for Industrial Communications (White Paper). Available online: https://5g-acia.org/whitepapers/integration-of-5g-with-time-sensitive-networking-for-industrial-communications/ (accessed on 22 November 2022).
115. Seijo, O.; Iturbe, X. Val I. Tackling the Challenges of the Integration of Wired and Wireless TSN with a Technology Proof-of-Concept. *IEEE Trans. Ind. Inform.* **2021**, *18*, 7361–7372. [CrossRef]
116. Chaine, P.J.; Boyer, M.; Pagetti, C.; Wartel, F. Egress-TT Configurations for TSN Networks. In Proceedings of the 30th International Conference on Real-Time Networks and Systems 2022, Paris, France, 7–8 June 2022; pp. 58–69.
117. Atallah, A.A.; Hamad, G.B.; Mohamed, O.A. Routing and scheduling of time-triggered traffic in time-sensitive networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4525–4534. [CrossRef]
118. Vlk, M.; Hanzálek, Z.; Brejchová, K.; Tang, S.; Bhattacharjee, S.; Fu, S. Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1 Qbv time-sensitive networks. *IEEE Trans. Commun.* **2020**, *68*, 7023–7038. [CrossRef]
119. Reusch, N.; Craciunas, S.S.; Pop, P. Dependability-aware routing and scheduling for Time-Sensitive Networking. *IET Cyber-Phys. Syst. Theory Appl.* **2022**, *7*, 124–146. [CrossRef]
120. Feng, Z.; Deng, Q.; Cai, M.; Li, J. Efficient Reservation-based Fault-Tolerant Scheduling for IEEE 802.1 Qbv Time-Sensitive Networking. *J. Syst. Archit.* **2022**, *123*, 102381. [CrossRef]
121. Xu, L.; Xu, Q.; Tu, J.; Zhang, J.; Zhang, Y.; Chen, C.; Guan, X. Learning-based Scalable Scheduling and Routing Co-design with Stream Similarity Partitioning for Time Sensitive Networking. *IEEE Internet Things J.* **2022**, *9*, 13353–13363. [CrossRef]
122. Xu, L.; Xu, Q.; Zhang, Y.; Zhang, J.; Chen, C. Co-design approach of scheduling and routing in time sensitive networking. In Proceedings of the 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Tampere, Finland, 9–12 June 2020; Volume 1, pp. 111–116.
123. Reusch, N.; Zhao, L.; Craciunas, S.S.; Pop, P. Window-based schedule synthesis for industrial IEEE 802.1 Qbv TSN networks. In Proceedings of the 2020 16th IEEE International Conference on Factory Communication Systems (WFCS), Porto, Portugal, 27–29 April 2020; pp. 1–4.
124. Alnajim, A.; Salehi, S.; Shen, C.C. Incremental path-selection and scheduling for time-sensitive networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.

125. *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*; IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements; IEEE: Piscataway, NJ, USA, 2018; pp. 1–208. [CrossRef]
126. Atallah, A.A.; Hamad, G.B.; Mohamed, O.A. Fault-resilient topology planning and traffic configuration for IEEE 802.1 Qbv TSN networks. In Proceedings of the 2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS), Platja d'Aro, Spain, 2–4 July 2018; pp. 151–156.
127. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*; IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications; IEEE: Piscataway, NJ, USA, 2020; pp. 1–421. [CrossRef]
128. Cavalcanti, D.; Bush, S.; Illouz, M.; Kronauer, G.; Regev, A.; Venkatesan, G. Wireless TSN–Definitions, Use Cases & Standards Roadmap. *Avnu Alliance* **2020**, 1–6. [CrossRef]
129. *IEEE Std 802.1CB-2017*; IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability; IEEE: Piscataway, NJ, USA, 2017; pp. 1–102. [CrossRef]