Period-Aware Schedulability Optimization for Time-Triggered Traffic in Time-Sensitive Networking

Yiqin Lu, *Member, IEEE*, Zhuoxing Chen, Jiancheng Qin, and Weiqiang Pan

***Abstract*—Time-Sensitive Networking (TSN) is a promising network technology in real-time industries, as it can precisely control the transmission time of periodic high-priority time-triggered (TT) traffic according to the IEEE 802.1Qbv standard. However, existing studies in TSN still lack enough attention to schedulability when allocating time slots to TT traffic, while schedulability is a critical metric for evaluating the TSN scheduling algorithms. In this paper, we address this problem by focusing on the periodicity of TT traffic. After analyzing the impact of time slot allocation strategy on schedulability, we propose a novel concept called prior-allocated (PriA) time slots. Then, we design and simplify the computational process of the PriA time slots with the help of abstract algebra. Furthermore, we propose a congruent time slot concentric mapping (CoTiSCoM) to visualize the simplified process. Based on the PriA time slots, we propose two specific allocation strategies to optimize schedulability for different scheduling algorithms: solver-based period-aware time slot allocation (PATSA-S) strategy and heuristic period-aware time slot allocation (PATSA-H) strategy. Extensive simulation experiments under offline scheduling and online scheduling demonstrate that, compared to existing strategies, our proposed time slot allocation strategies significantly improve schedulability with an acceptable runtime overhead.**

***Index Terms*—Deterministic communications, time-sensitive networking (TSN), IEEE 802.1Qbv, schedulability, abstract algebra.**

I. Introduction

In RECENT years, numerous real-time industries, such as industrial control systems, power systems, and autonomous vehicles, have rapidly evolved toward intelligence. During this transformation, these industries require the communication networks to reliably transmit periodic high-priority traffic with bounded latency and jitter, i.e., in a deterministic manner. However, the various traditional network protocols within a single real-time industry are often tightly coupled with specific equipment manufacturers. This coupling, along with a wide range of physical interfaces, results in communication conflicts and fragmentation of physical connection protocols [1].

In response to the urgent demand for a flexible and unified network in real-time industries, the IEEE Time-Sensitive Networking (TSN) task group [2] has proposed a set of Ethernet extension standards to meet the diverse real-time requirements. Currently, TSN has emerged as a promising network technology for real-time domains and is gradually moving towards large-scale integrated networks [3], [4].

Among the TSN standards, IEEE 802.1Qbv [5] stands out as one of the core specifications. It can precisely control the transmission time of high-priority traffic, achieving zero-jitter transmission and deterministic communications. Specifically, IEEE 802.1Qbv regulates that each egress port of network devices uses specific queues to transmit high-priority traffic. Furthermore, it introduces Gate Control Lists (GCLs) to precisely manage the opening and closing states of these queues. A high-priority stream can only pass through its assigned queue when the corresponding gate is open. Due to its time-triggered scheduling mechanism, the periodic high-priority traffic is referred to as time-triggered (TT) traffic in TSN.

Under the IEEE 802.1Qbv standard, the key issue is researching the specific scheduling algorithms to compute the GCLs, i.e., the exact transmission time slots to transmit TT traffic, where the time slot is the minimum time unit used in network devices. For the numerous research results related to scheduling algorithms [6], [7], schedulability is a critical metric to evaluate their performance. Specifically, schedulability is defined as the ratio of instances where all TT streams can be successfully scheduled to the total number of instances [8], which quantifies the ability of an algorithm to schedule TT traffic across various real-time scenarios. Other performance metrics, such as end-to-end latency, flowspan, and scalability, are only meaningful when the TT traffic is schedulable.

In practice, the schedulability of an algorithm can be affected by its different phases, such as routing, constraint construction, and time slot allocation. Unfortunately, the existing research based on the IEEE 802.1Qbv standard still lacks theoretical research on improving schedulability by optimizing the time slot allocation strategy, which is the fundamental component of scheduling algorithms. The time slot allocation strategy determines how specific time slots are allocated to each TT stream in the final phase of scheduling algorithms. In TSN, allocating the time slots to each TT stream has a profound impact on schedulability. For example, in online scheduling scenarios, the inappropriately allocated time slots of scheduled streams may cause newly arrived TT streams to be unschedulable, resulting in complex reconfiguration problems.

In this paper, we study how to optimize the time slot allocation strategy through mathematical analysis to improve the

Received 24 October 2024; revised 14 May 2025; accepted 7 July 2025. Date of publication 15 July 2025; date of current version 21 November 2025. This work was supported by the National Key Research and Development Program of China under Grant 2020YFB1805300. Recommended for acceptance by Dr. S. Silvestri. *(Corresponding author: Zhuoxing Chen.)*

The authors are with the School of Electronics and Information Engineering, South China University of Technology (SCUT), Guangzhou 510640, China (e-mail: eeyqlu@scut.edu.cn; eeczx@mail.scut.edu.cn; jcqin@scut.edu.cn; wqpan@scut.edu.cn).

Digital Object Identifier 10.1109/TNSE.2025.3589270

schedulability. Since the TT traffic is periodically transmitted, each TT stream needs to be allocated periodic time slots. However, the allocated time slots for each TT stream typically belong to multiple residue classes modulo other periods. As a result, the scheduled TT streams reduce the solution space of the unscheduled TT streams. When all residue classes modulo a specific period contain allocated time slots, the unscheduled TT streams with that period are unschedulable. Therefore, we focus on periodicity, the core attribute of TT traffic, and propose a novel concept called prior-allocated (PriA) time slots. Leveraging abstract algebra, we design and simplify the computational process of PriA time slots. Furthermore, we propose two specific strategies to realize it.

The main contributions of this paper are listed as follows.

- We point out that prioritizing the allocation of time slots congruent with the allocated time slots is helpful in scheduling more TT streams, which provides a new perspective on optimizing the schedulability in TSN. To the best of our knowledge, this is a novel study on improving the schedulability of TT traffic by optimizing the time slot allocation strategy.
- We conduct a theoretical analysis of the impact of time slot allocation strategy on schedulability. On this basis, we design and simplify a novel prior-allocated (PriA) time slots leveraging abstract algebra. The group theory and congruence contained in abstract algebra are well suited to the periodicity of TT traffic. Furthermore, to visualize the computational process of PriA time slots intuitively, we propose a congruent time slot concentric mapping (CoTiSCoM).
- Based on the proposed PriA time slots and CoTiSCoM, we propose a solver-based period-aware time slot allocation (PATSA-S) strategy and a heuristic period-aware time slot allocation (PATSA-H) strategy for corresponding types of scheduling algorithms. They can efficiently exploit the solution space to optimize the schedulability in both offline and online scheduling scenarios. In addition, the PATSA-S and PATSA-H strategies are adaptable to most existing scheduling algorithms.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the architecture model, the application model, and the time slot model of TSN. Section IV explains the motivation behind our work, analyzes the impact of time slot allocation strategy on schedulability, and then introduces and simplifies the PriA time slots. On this basis, Section V describes the proposed PATSA-S and PATSA-H strategies. Section VI presents and discusses the simulation results of the proposed strategies. Finally, Section VII concludes this paper and outlines future work.

II. Related Work

In this section, we first review the related work on schedulability optimization. Subsequently, we present the existing time slot allocation strategies based on the IEEE 802.1Qbv standard, which have a significant impact on schedulability.

A. Schedulability Optimization

Since schedulability reflects the core performance of TSN scheduling algorithms, its optimization is an important topic in TSN scheduling [9], [10]. Existing research on improving schedulability has largely focused on optimizing the routing of TT traffic. For example, Chang et al. [11] propose a Real-Time Routing Scheduler (RTRS). They route each TT stream with the worst-case end-to-end delay (WCED) value among all feasible paths to ensure that all streams are schedulable. With the help of reinforcement learning, Min et al. [12] propose a learning-based routing method for TT traffic to improve load-balanced routes for higher schedulability. Besides, [13], [14], [15] are some other representative works on optimizing routing to improve schedulability. However, the candidate paths introduced in the routing optimization will significantly increase the solution space and search overhead.

The stream partitioning scheduling emerging in recent years can significantly improve the computational efficiency of TT traffic scheduling [8], [16], [17]. In particular, in our previous work [17], we proposed a dynamic stream partitioning (DSP) method and designed a partitioning-aware dynamic routing (PADR) based on it. According to the scheduling solution of the partitioned streams, the PADR can improve the schedulability of TT traffic by dynamically adjusting the routing of unpartitioned TT streams. However, while [17] focuses on optimizing the stream partitioning process, how to optimize the schedulability of TT traffic after partitioning from the perspective of time slot allocation is still an open issue.

In addition, schedulability is crucial in online scheduling, where newly arrived TT streams can be dynamically added to the network at any time. Prevailing methods to improve the schedulability mainly focus on rescheduling the scheduled streams that conflict with the newly arrived streams [18], [19], [20], [21]. For example, Nasrallah et al. [18] improve the schedulability of TT traffic at the expense of delay increases for low-priority best-effort (be) traffic, and design a complete reconfiguration process with respect to network deployment parameters. In large-scale networks, frequent rescheduling or parameter reconfiguration introduces huge additional overhead, which ideally should be avoided.

Moreover, [1], [22] provide another interesting solution to improve schedulability. They relax the classical scheduling constraints to expand the solution space. However, this solution requires hardware enhancements to ensure the deterministic transformation of TT traffic.

B. Time Slot Allocation Strategies Based on IEEE 802.1Qbv

The scheduling algorithms based on the IEEE 802.1Qbv standard are classified into exact approaches and heuristic approaches [6], [23]. Correspondingly, the time slot allocation strategies contained in these algorithms are also classified into two types: solver-based allocation strategies and heuristic allocation strategies. Both types of allocation strategies follow the scheduling constraints proposed in [24], ensuring conflict-free scheduling of TT traffic. Specifically, solver-based allocation

strategies employ Satisfiability Modulo Theories (SMT) or Integer Linear Programming (ILP) solvers to randomly allocate feasible time slots to TT traffic within the entire solution space [8], [10], [15]. However, these strategies cannot accurately compute the allocated time slots, making it challenging to optimize schedulability in online scheduling scenarios. In contrast, heuristic allocation strategies can accurately allocate the specific time slots to each TT stream based on predefined rules. The most common heuristic strategies are as soon as possible (ASAP) and as late as possible (ALAP), which allocate the earliest or latest available time slots to the current TT stream, respectively [25], [26]. Although these heuristic strategies can rapidly allocate the time slots, they only explore a limited portion of the solution space according to the predefined rules, resulting in a degradation of schedulability.

In addition, to reduce the length of GCLs in TSN switches, Dürr et al. [27] propose the no-wait scheduling rule. This rule regulates that the TT streams are forwarded immediately upon arrival at TSN switches, which can be included as a constraint in the scheduling solvers [16], as well as be implemented in scheduling algorithms as a no-wait allocation strategy [28]. However, when searching for scheduling results, this rule introduces an unnecessary restriction that reduces the schedulability of TT traffic [23].

Overall, existing research on schedulability optimization either introduces significant overhead or is limited to specific scheduling scenarios. More importantly, the optimization of time slot allocation strategies underlying scheduling algorithms has still been overlooked in terms of schedulability. In scenarios where not all TT streams are scheduled simultaneously, both solver-based and heuristic time slot allocation strategies have dynamic impacts on the schedulability.

III. System Model

A. Architecture Model

In our work, we introduce a directed graph *G(V, E)* to model the architecture of TSN, where *V* represents the set of TSN devices and *E* represents the set of physical links between the devices. In the graph *G*, the TSN devices include switches and end stations, and each egress port of each device v_a ($v_a \in V$) has eight queues to transmit traffic with different priorities. In addition, all physical links are full-duplex, and we define $[v_a, v_b] \in E$ to denote the directed link from network device v_a to v_b ($v_a, v_b \in V$). Moreover, to efficiently compute and distribute the GCLs, we adopt the fully centralized architecture specified in the IEEE 802.1Qcc standard [29]. Furthermore, to accurately execute the scheduling results, the TSN devices in *G* follow the IEEE 802.1AS standard [30] and synchronize the time with high precision.

B. Application Model

The various real-time applications running in TSN will periodically generate TT streams at end stations. In this paper, we define $S = \{s_1, s_2, \ldots, s_m\}$ as the set of TT streams transmitted within topology *G*, where *m* denotes the number of TT streams.

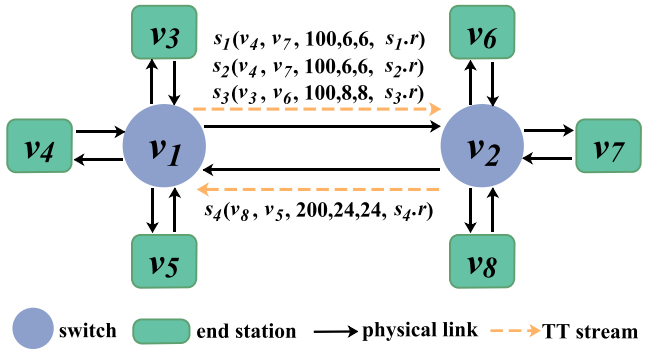

Fig. 1. An illustrative example of TSN topology and traffic.

In addition, each TT stream $s_i \in S$ can be modeled by a six-tuple property $(s_i.sen, s_i.rec, s_i.size, s_i.p, s_i.d, s_i.r)$. Specifically, $s_i.sen$ and $s_i.rec$ denote the sender and receiver of s_i, respectively. $s_i.size$ represents the data size of s_i, i.e., the number of bytes generated in each period of s_i. In our study, $s_i.size$ does not exceed a maximum transmission unit (MTU). Besides, $s_i.p$ denotes the period of s_i, which is a core attribute of TT streams in industrial scenarios. We introduce $P = \{p_1, p_2, \ldots, p_n\}$ to denote the set of different periods of TT streams, where *n* denotes the number of periods. For these periods, we refer to the least common multiple of them as the hyper-period and denote it by the notation *hper*. Since each stream $s_i \in S$ is transmitted periodically, the GCLs repeat cyclically with a period of *hper*. Moreover, $s_i.d$ denotes the deadline of s_i, and $s_i.d \leq s_i.p$. We can ensure the real-time requirement of s_i if it can arrive at $s_i.rec$ before $s_i.d$. Furthermore, the last element $s_i.r$ in the six-tuple property denotes the path of s_i, which is calculated by the classical shortest path routing in this paper.

Fig. 1 shows an example of a simple TSN topology consisting of two TT switches, each connected to three end stations. This basic structure can form the complex topologies in different scenarios. Fig. 1 also includes four TT streams with known attributes, and their paths are determined by the shortest path routing.

C. Time Slot Model

The calculations involved in this paper for time slots are on the numerical values of their indices, and we number the time slots on each link separately starting from 0. Therefore, the set of time slots within the hyper-period is $\{0, 1, \ldots, hper - 1\}$. In addition, assuming that the period of each stream is an integer, we can classify all the time slots within $hper$ into $p_j \in P$ sets as follows: if the remainder of a given time slot is a when dividing p_j, then it belongs to the set a. Since $hper$ is a multiple of p_j, all time slots within $hper$ can be uniquely classified into a specific set. We introduce the notation $[a]_{p_j}$ to denote the set of congruent time slots that leave the same remainder as a when divided by p_j. In abstract algebra, $[a]_{p_j}$ represents the residue class of a modulo p_j, and defined as

$$[a]_{p_j} = \{b \mid b \equiv a \pmod{p_j}\}, a \in \{0, 1, \ldots, p_j - 1\}, \quad (1)$$

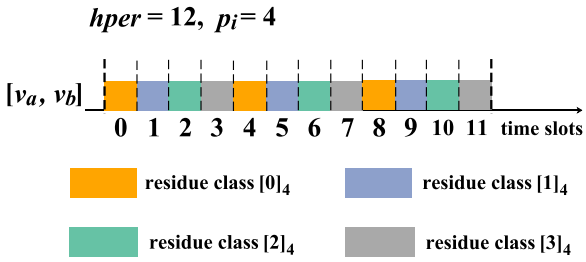

Fig. 2. An illustrative example of the time slot model: Classification of time slots into different residue classes modulo 4 within $hper$ =12.

where '$\equiv$' denotes the congruence relation, i.e., $(b-a)$ is divisible by integer p_j.

In order to further explain residue class and congruence, we provide a simple example in Fig. 2, where $hper$ =12 and p_j =4. We use different colors to represent different residue classes and show the time slots with congruence relation in the same color. For example, time slots 2 and 6 both leave the same remainder as 2 when divided by 4, so they belong to residue class $[2]_4$ and are colored green.

Furthermore, we introduce the notation $s_i.len$ to denote the length of consecutive time slots allocated to s_i within $s_i.p$, representing the transmission delay of s_i. For the sake of simplicity, we assume the bandwidth of TT traffic on each link is the same. Therefore, $s_i.len = \frac{s_i.size}{bandwidth}$. Considering $s_i.size \leq$ MTU (1500B), we introduce maxlen $= \frac{MTU}{bandwidth}$ to denote the maximum value of $s_i.len$ among all TT streams. Moreover, we introduce $s_i^{[v_a,v_b]}.t$ to denote the first time slot allocated to s_i within its first period on link $[v_a, v_b]$. Once $s_i.len$ and $s_i^{[v_a,v_b]}.t$ are calculated, the time slots allocated to s_i in subsequent periods are periodically repeated.

For ease of reference, the notations used throughout the paper are listed in Table I.

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
$G(V, E)$	the graph to model the network architecture
V	the set of TSN devices
E	the set of directed physical links
v_a, v_b	the TSN device a, b, respectively
$[v_a, v_b]$	a directed link from v_a to v_b
S	the set of TT streams
m	the number of TT streams
s_i	the TT stream i
$s_i.sen$	the sender of s_i
$s_i.rec$	the receiver of s_i
$s_i.size$	the data size of s_i
$s_i.p$	the period of s_i
P	the set of different periods of TT streams
n	the number of periods
p_j	the period j
$hper$	the least common multiple of TT traffic periods
$s_i.d$	the deadline of s_i
$s_i.r$	the path of s_i
$[a]_{p_j}$	the set of congruent time slots that leave the same remainder of a when divided by p_j
$s_i.len$	the length of consecutive time slots allocated to s_i
$maxlen$	the maximum value of $s_i.len$ among all streams
$s_i^{[v_a,v_b]}.t$	the first time slot allocated to s_i on link $[v_a, v_b]$
α_i	the number of $s_i.p$ within $hper$
T_i^A	the set of time slots allocated to s_i within $hper$
T^*	the set of prior-allocated time slots
$T_{p_j}^*$	the set of prior-allocated time slots for the TT streams with period p_j
R_i	the set of residue classes that contain time slots allocated to s_i
R_{i,p_k}	the set of residue classes modulo p_k that contain time slots allocated to s_i
$\Delta(T_{p_j}^*)^i$	the set of new prior-allocated time slots added to $T_{p_j}^*$ as a result of the allocation to s_i
$\Delta(T_{p_j}^*)_{p_k}^i$	a subset of $\Delta(T_{p_j}^*)^i$ to optimize the schedulability of the TT streams with p_k
G_{p_j}	the set of all residue classes modulo p_j
G_{p_j,p_k}	the subset of G_{p_j} derived from the multiples of p_k
$\Delta'(T_{p_j}^*)^{s_i.p}$	the baseline for $\Delta(T_{p_j}^*)^i$
$\Delta'(T_{p_j}^*)_{p_k}^{s_i.p}$	the baseline for $\Delta(T_{p_j}^*)_{p_k}^i$

IV. Time Slot Allocation for Schedulability Optimization: Analysis and Design

In this section, we first explain the motivation behind our work through an intuitive example. We then prove that the time slots allocated to a given TT stream can affect the schedulability of the other TT streams. Building on this foundation, we propose a novel concept called prior-allocated (PriA) time slots, and simplify the computational process of PriA time slots leveraging abstract algebra. Furthermore, we propose the congruent time slot concentric mapping (CoTiSCoM) for their intuitive representation.

A. Motivation

Different time slot allocation strategies typically allocate different time slots to each TT stream, resulting in different impacts on the schedulability. Consider an intuitive example: in the topology shown in Fig. 2, TT streams $\{s_1, s_2, s_3\}$ are transmitted under different time slot allocation strategies. We define the time slots starting from 0 and specify that each time slot can transmit 100B data. Since the processing delay in each device and the propagation delay on each link are usually much smaller than the transmission delay in TSN, we set them to 0 to simplify our example. In addition, We set the scheduling order of streams $\{s_1, s_2, s_3\}$ as follows: first s_1, then s_2, and finally s_3.

Fig. 3(a) and (b) show the scheduling results under the ASAP and ALAP strategies, respectively. In both cases, s_3 conflicts with s_1 on link $[v_1, v_2]$. Even worse, under both ASAP and ALAP strategies, s_3 is unschedulable with a period of 8 on link $[v_1, v_2]$, regardless of how the time slots allocated to it are adjusted.

To analyze the underlying reasons, we mark the residue classes modulo 8 occupied by the time slots of TT streams on $[v_1, v_2]$. As shown in Fig. 3(a) and (b), all the residue classes modulo 8 have time slots allocated to s_1 and s_2. Consequently, s_3 cannot be allocated time slots with a period of 8.

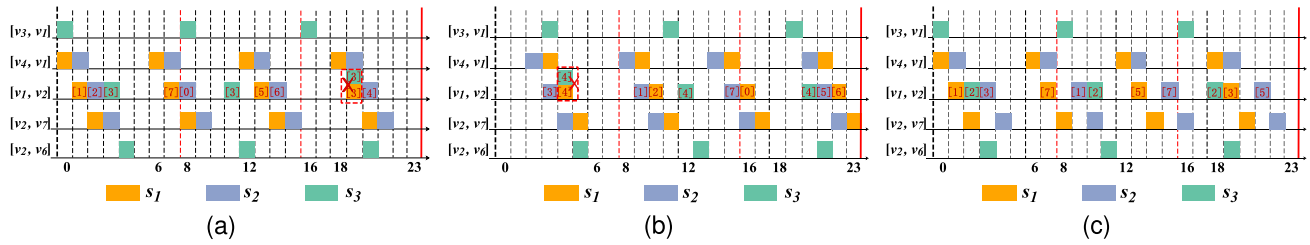

Fig. 3. The scheduling results for TT streams scheduled in order s_1, s_2, and s_3 under different time slot allocation strategies. (a) Under ASAP strategy. (b) Under ALAP strategy. (c) Allocating s_2 to the time slots congruent to the allocated time slots of s_1 modulo 8.

Following this observation, we adjust the time slots allocated to s_2, which are congruent to the allocated time slots of s_1 modulo 8. Specifically, considering the ASAP strategy, the time slots allocated to s_1 on link $[v_1, v_2]$ are 1, 7, 13, 19, which belong to residue classes $[1]_8$, $[7]_8$, $[5]_8$, $[3]_8$, respectively. Therefore, we adjust $s_2^{[v_1,v_2]}.t$ to 3, and leave residue classes$[0]_8$, $[2]_8$, $[4]_8$, $[6]_8$ to s_3. Fig. 3(c) shows that streams $\{s_1, s_2, s_3\}$ are all schedulable under this optimization.

This example reveals that the periodicity of TT traffic is closely related to scheduling results, and a period-aware time slot allocation strategy can optimize the schedulability. It is worth noting that the solver-based time slot strategies can only ensure scheduling s_1, s_2, and s_3 when allocating time slots to them simultaneously. However, in online scheduling scenarios, if one of these streams is newly arrived and scheduled separately, it may become unschedulable.

B. Schedulability Analysis Focusing on Time Slot Allocation

Implementing the insights from the example elaborated in the motivation requires further analysis and careful design. We first introduce notation $\alpha_i = \frac{hper}{s_i.p}$ to denote the number of periods of s_i within $hper$. With the help of abstract algebra, the specific analysis of how the time slot allocation strategy affects schedulability is as follows.

Lemma 1: A necessary condition for an arbitrary TT stream $s_i \in S$ ($s_i.len = x$, and $x \in \{1, 2, \ldots, maxlen\}$) to be schedulable on the link $[v_a, v_b] \in E$ is as follows: there exist x consecutive residue classes modulo $s_i.p$ on link $[v_a, v_b]$ such that all congruent time slots belonging to these residue classes can be allocated to s_i.

Proof: We first denote the set of time slots allocated to an arbitrary TT stream $s_i \in S$ within *hper* as T_i^A. Based on the periodicity of TT traffic, when $s_i.len = x$ ($x \in \{1, 2, \ldots, maxlen\}$), T_i^A is formulated as follows:

$$\begin{aligned} T_i^A = \{ & s_i^{[v_a,v_b]}.t, s_i^{[v_a,v_b]}.t + s_i.p, \ldots, s_i^{[v_a,v_b]}.t + (\alpha_i - 1) * s_i.p, \\ & s_i^{[v_a,v_b]}.t + 1, \ldots, s_i^{[v_a,v_b]}.t + 1 + (\alpha_i - 1) * s_i.p, \\ & \ldots, \\ & s_i^{[v_a,v_b]}.t + x - 1, \ldots, s_i^{[v_a,v_b]}.t + x - 1 + (\alpha_i - 1) * s_i.p \}. \end{aligned} \tag{2}$$

Dividing the time slots in T_i^A by $s_i.p$, we can find the set T_i^A consists of x consecutive residue classes modulo $s_i.p$ from $[s_i^{[v_a,v_b]}.t]_{s_i.p}$ to $[s_i^{[v_a,v_b]}.t + x - 1]_{s_i.p}$. Moreover, each residue class modulo $s_i.p$ contains α_i congruent time slots. Therefore, T_i^A is composed of all congruent time slots belonging to x consecutive residue classes modulo $s_i.p$, and the absence of any of these time slots would make s_i unschedulable.

Furthermore, when all time slots belonging to a residue class $[a]_{p_j}$ can be allocated to the TT streams with period p_j, we define $[a]_{p_j}$ as an allocatable reside class modulo p_j. On this basis, we have Theorem 1 as follows.

Theorem 1: When scheduling an arbitrary TT stream $s_i \in S$ on $[v_a, v_b] \in E$ ($s_i.len = x$, and $x \in \{1, 2, \ldots, \text{maxlen}\}$), as the number of allocatable residue classes modulo $s_i.p$ decreases, the upper bound of probability that s_i is schedulable on $[v_a, v_b]$ will also decrease.

Proof: See Appendix A.

In conclusion, this analysis demonstrates the root cause of why TT streams are unschedulable in TSN. Specifically, an arbitrary stream s_i is unschedulable on link $[v_a, v_b]$ if the number of consecutive allocatable residue classes modulo $s_i.p$ is less than $s_i.len$. Therefore, the time slot allocation strategy determines the specific time slots allocated to the TT streams, which can dynamically affect the number of allocatable residue classes and the schedulability of TT traffic.

C. Prior-Allocated Time Slots Design

Based on the analysis, we propose the concept of prior-allocated (PriA) time slots, which are prioritized for allocation to TT streams within their first periods. Specifically, we introduce the notation T^* to denote the set of PriA time slots for the unscheduled TT streams. T^* is formulated as follows:

$$T^* = T_{p_1}^* \cup T_{p_2}^* \cup \cdots \cup T_{p_n}^*, \tag{3}$$

where $T_{p_j}^*$ ($p_j \in P$) denotes the set of PriA time slots for the TT streams with period p_j.

In order to maximize the number of allocatable residue classes modulo an arbitrary period $p_k \in P$, we prioritize the allocation of time slots that are congruent with the allocated time slots modulo p_k to the TT streams with different period p_j ($p_j \in P$ and $p_j \neq p_k$). Therefore, we first compute the allocated residue classes after each allocation, and then dynamically update the PriA time slots.

Specifically, after allocating time slots to an arbitrary TT stream $s_i \in S$, we introduce notation R_i to denote the set of residue classes that the allocated time slots belong to. R_i is

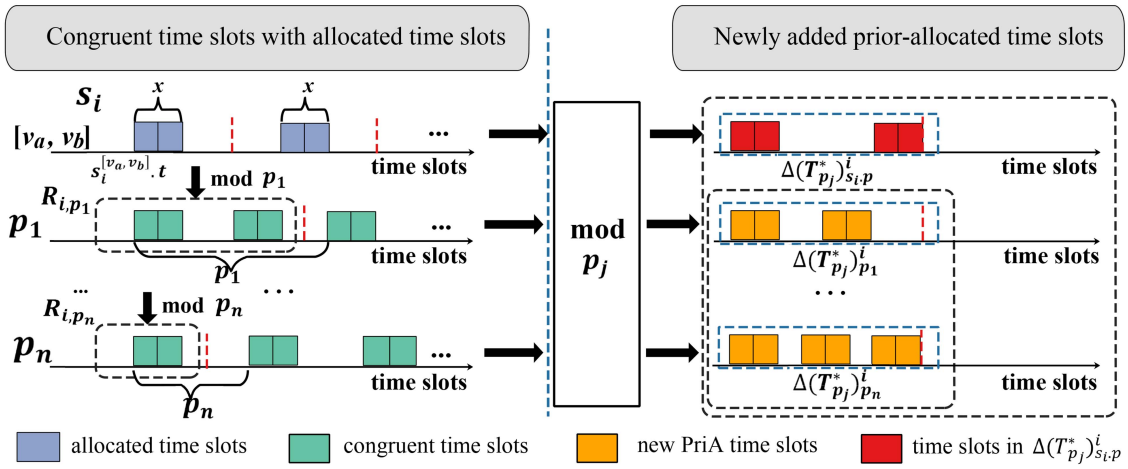

Fig. 4. Computation process of $\Delta(T^*_{p_j})^i$ after allocating time slots to an arbitrary TT stream s_i on the link $[v_a, v_b] \in s_i.r$.

formulated as follows:

$$R_i = R_{i,p_1} \cup R_{i,p_2} \cup \cdots \cup R_{i,p_n}, \tag{4}$$

where each set R_{i,p_k} ($p_k \in P$) denotes the residue classes modulo period p_k that contain time slots allocated to s_i. Based on (20), R_{i,p_k} is formulated as follows:

$$\begin{aligned} R_{i,p_k} &= \left\{ \left[t_i^A \bmod p_k\right]_{p_k} \mid t_i^A \in T_i^A \right\} \\ &= \Big\{ [t \bmod p_k]_{p_k}, [(t + s_i.p) \bmod p_k]_{p_k}, \ldots, \\ &\quad [(t + (\alpha_i - 1) * s_i.p) \bmod p_k]_{p_k}, \ldots, \\ &\quad [(t+x-1) \bmod p_k]_{p_k}, [(t+x-1+s_i.p) \bmod p_k]_{p_k} \\ &\quad \cdots, [(t + x - 1 + (\alpha_i - 1) * s_i.p) \bmod p_k]_{p_k} \Big\}, \\ t &= s_i^{[v_a,v_b]}.t. \end{aligned} \tag{5}$$

Furthermore, after allocating time slots to s_i, we update $T^*_{p_j}$ as follows:

$$T^*_{p_j} = T^*_{p_j} \cup \Delta(T^*_{p_j})^i, \tag{6}$$

where $\Delta(T^*_{p_j})^i$ denotes the set of new PriA time slots added to $T^*_{p_j}$. $\Delta(T^*_{p_j})^i$ consists of multiple subsets that can optimize the schedulability of the unscheduled streams with different periods. We denote each of these subsets as $\Delta(T^*_{p_j})^i_{p_k}$ ($p_k \neq p_j$), and $\Delta(T^*_{p_j})^i$ is formulated as follows:

$$\Delta(T^*_{p_j})^i = \Delta(T^*_{p_j})^i_{p_1} \cup \cdots \cup \Delta(T^*_{p_j})^i_{p_n} - \Delta(T^*_{p_j})^i_{s_i.p}. \tag{7}$$

To compute $\Delta(T^*_{p_j})^i_{p_k}$ ($p_k \neq s_i.p$), we first calculate the congruent time slots for each residue class in R_{i,p_k} within $hper$, and then take these time slots modulo p_j. Therefore, $\Delta(T^*_{p_j})^i_{p_k}$ is represented as:

$$\Delta(T^*_{p_j})^i_{p_k} = \left\{ t_{i,p_k} \bmod p_j \middle|\, t_{i,p_k} \in r_{i,p_k}, r_{i,p_k} \in R_{i,p_k} \right\}. \tag{8}$$

In addition, when $p_k = s_i.p$, $\Delta(T^*_{p_j})^i_{s_i.p}$ is represented as:

$$\Delta(T^*_{p_j})^i_{s_i.p} = \left\{ t_{i,s_i.p} \bmod p_j \middle|\, t_{i,s_i.p} \in r_{i,s_i.p}, r_{i,s_i.p} \in R_{i,s_i.p} \right\}$$

$$= \left\{ t_i^A \bmod p_j \mid t_i^A \in T_i^A \right\}. \tag{9}$$

For ease of understanding, Fig. 4 visualizes the computation process of the set of newly added time slots $\Delta(T^*_{p_j})^i$ for the period $p_j \in P$ after allocating time slots to an arbitrary TT stream s_i on the link $[v_a, v_b] \in s_i.r$. Specifically, Fig. 4 shows the key variables involved in (4)–(9) and colors the set $\Delta(T^*_{p_j})^i$ in orange.

D. Simplification of the Design

We have elaborated on the intuitive design of PriA time slots. However, the implementation of this design may be complex. The main reason is that the allocated time slots are different for each TT stream, which leads to complex iterative computations when dynamically updating $\Delta(T^*_{p_j})^i$ according to (4)–(9). Fortunately, we can simplify our design with the help of abstract algebra. The connection between group theory and time slots is described below.

Lemma 2: For the time slots within $hper$, the set of all residue classes modulo an arbitrary period $p_j \in P$ forms a finite cyclic group G_{p_j}.

Proof: See Appendix B.

Theorem 2: For any $p_k, p_j \in P$, and given the set of time slots $T_{p_k} = \{i * p_k \mid 0 \leq i \leq \frac{hper}{p_k} - 1\}$. Define the set of residue classes $G_{p_j,p_k} = \{[t_{p_k} \bmod p_j]_{p_j} \mid t_{p_k} \in T_{p_k}\}$. Then, G_{p_j,p_k} is a subgroup of group G_{p_j}. Furthermore, G_{p_j,p_k} is a finite cyclic group, and $[gcd(p_j, p_k)]_{p_j}$ is a generator of G_{p_j,p_k}, where $[gcd(p_j, p_k)]_{p_j}$ denotes the greatest common divisor of p_j and p_k.

Proof: See Appendix C.

Since the above mathematical analysis is independent of the specific allocation process, we can uniformly compute G_{p_j,p_k} between any two periods during the initialization of the TSN, and then compute the baseline for updating PriA time slots. Subsequently, during the update process, we only need to calculate the offsets to the baseline caused by the specific $s_i^{[v_a,v_b]}.t$.

Specifically, during network initialization, we assume $s_i^{[v_a,v_b]}.t = 0$, and simplify the allocated residue classes of period $p_k \in P$ as:

$$R_{i,p_k} = G_{p_k,s_i.p} = \left\{ [0]_{p_k}, [1*g]_{p_k}, \ldots, \left[\left(\frac{p_k}{g} - 1 \right) * g \right]_{p_k} \right\}, \quad g = gcd(p_k, s_i.p). \tag{10}$$

In addition, we introduce the set $\Delta'(T^*_{p_j})^{s_i.p}$ to represent the baseline for $\Delta(T^*_{p_j})^i$. It is formulated as follows:

$$\Delta'(T^*_{p_j})^{s_i.p} = \Delta'(T^*_{p_j})^{s_i.p}_{p_1} \cup \cdots \cup \Delta'(T^*_{p_j})^{s_i.p}_{p_n} - \Delta'(T^*_{p_j})^{s_i.p}_{s_i.p}, \tag{11}$$

where $\Delta'(T^*_{p_j})^{s_i.p}_{p_k}$ $(p_k \neq s_i.p)$ and $\Delta'(T^*_{p_j})^{s_i.p}_{s_i.p}$ serve as the baselines for $\Delta(T^*_{p_j})^i_{p_k}$ and $\Delta(T^*_{p_j})^i_{s_i.p}$ when $s_i^{[v_a,v_b]}.t = 0$, respectively. They are respectively formulated as follows:

$$\Delta'(T^*_{p_j})^{s_i.p}_{p_k} = \left\{ ((g_{p_k,s_i.p})^{\min} + G_{p_j,p_k}) \bmod p_j \mid g_{p_k,s_i.p} \in G_{p_k,s_i.p} \right\}, \tag{12}$$

$$\Delta'(T^*_{p_j})^{s_i.p}_{s_i.p} = \left\{ (g_{p_j,s_i.p})^{\min} \mid g_{p_j,s_i.p} \in G_{p_j,s_i.p} \right\}, \tag{13}$$

where $(.)^{\min}$ denotes the minimum time slot of the corresponding residue class, i.e., $([a]_{p_j})^{\min} = a$.

After allocating time slots to s_i $(s_i^{[v_a,v_b]}.t \geq 0)$, we offset $\Delta'(T^*_{p_j})^{s_i.p}_{p_k}$ and simplify $\Delta(T^*_{p_j})^i_{p_k}$ in the update process as follows:

$$\Delta(T^*_{p_j})^i_{p_k} = \left\{ \left(s_i^{[v_a,v_b]}.t + x + \Delta'(t^*_{p_j})^{s_i.p}_{p_k} \right) \bmod p_j \mid x \in \{1,2,\ldots,s_i.len\}, \Delta'(t^*_{p_j})^{s_i.p}_{p_k} \in \Delta'(T^*_{p_j})^{s_i.p}_{p_k} \right\}. \tag{14}$$

When $p_k = s_i.p$, $\Delta(T^*_{p_j})^i_{s_i.p}$ can also be calculated by (14).

In summary, the simplified design uniformly quantifies the impact of different periods on schedulability during network initialization. In addition, the simplified (10)–(14) replace the complex traversal of T_i^A in (20), (5), (8) and (9) with corresponding subgroups. These subgroups can efficiently be calculated according to Theorem 2. For example, when $P = \{20, 30, 60, \cdots\}$, and $s_i.p = 30$, $p_j = 60$, and $p_k = 20$, it is straightforward to compute that $gcd(p_k, s_i.p) = 10$ and $gcd(p_j, s_i.p) = 30$. According to (12) and (13), $\Delta'(T^*_{p_j})^{s_i.p}_{p_k} = \{[0]_{60}, [10]_{60}, [20]_{60}, [30]_{60}, [40]_{60}, [50]_{60}\}$ and $\Delta'(T^*_{p_j})^{s_i.p}_{s_i.p} = \{[0]_{60}, [30]_{60}\}$. During the update process, when $s_i^{[v_a,v_b]}.t = 5$ and $s_i.len = 1$, $\Delta(T^*_{p_j})^i_{p_k} = \{5, 15, 25, 35, 45, 55\}$ and $\Delta(T^*_{p_j})^i_{s_i.p} = \{5, 35\}$ according to (14). Moreover, when $s_i^{[v_a,v_b]}.t = 18$ and $s_i.len = 2$, $\Delta(T^*_{p_j})^i_{p_k} = \{18, 19, 28, 29, 38, 39, 48, 49, 58, 59, 8, 9\}$ and $\Delta(T^*_{p_j})^i_{s_i.p} = \{18, 19, 48, 49\}$.

E. Congruent Time Slot Concentric Mapping

In order to visualize the simplified computational process of PriA time slots, we introduce a congruent time slot concentric mapping (CoTiSCoM). CoTiSCoM consists of multiple sub-mappings, where each sub-mapping consists of multiple concentric circles and can map congruent time slots between different periods. Without loss of generality, we assume that for an arbitrary period $p_j \in P$, there exist other periods $P' \subseteq P$, and each period contained in P' shares the same greatest common divisor with p_j. We order these periods in ascending order, with p_j being larger than k of them.

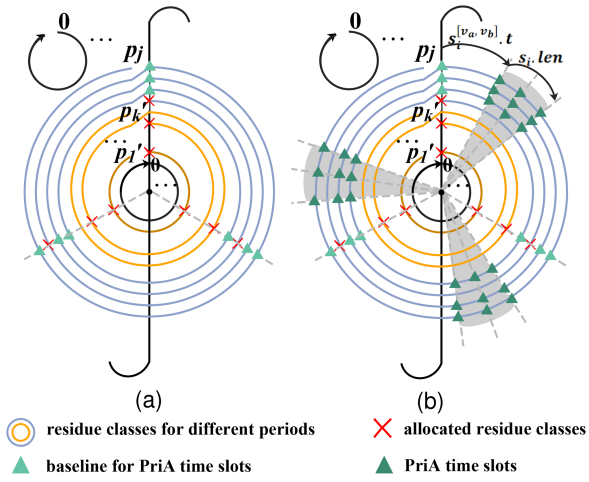

Fig. 5. An illustrative example of CoTiSCoM, which visualizes the computation of $\Delta(T^*_{p_j})^i$. (a) During network initialization, mark R_i on the concentric circles, then mark $\Delta'(T^*_{p_j})^{s_i.p}$ according to R_i. (b) After allocating the time slots to s_i, rotate $\Delta'(T^*_{p_j})^{s_i.p}$ clockwise by $s_i^{[v_a,vb]}.t$ units to obtain the corresponding $\Delta(T^*_{p_j})^i$.

Fig. 5 illustrates how CoTiSCoM visualizes the computation of $\Delta(T^*_{p_j})^i$. Specifically, an arbitrary period $p'_k = \beta * gcd(p_j, p'_k)$ is represented as β concentric circles, and each circle contains $gcd(p_j, p'_k)$ residue classes. Besides, the residue classes modulo p'_k are counted clockwise from 0 to $(p'_k - 1)$.

As shown in Fig. 5(a), during network initialization, we first compute the allocated residue classes of each period when $s_i^{[v_a,v_b]}.t = 0$, and mark these residue classes in the corresponding concentric circles with red crosses. We then map these residue classes toward p_j and pass through each concentric circle contained in p_j. The intersections, excluding those indicated by red crosses, are exactly the baseline $\Delta'(T^*_{p_j})^{s_i.p}$ for $\Delta(T^*_{p_j})^i$ and are marked with light green triangles.

In addition, as shown in Fig. 5(b), after allocating the time slots to an arbitrary TT stream with $s_i.p = p_j$ and $s_i.len = x$, we rotate $\Delta'(T^*_{p_j})^{s_i.p}$ clockwise by $s_i^{[v_a,vb]}.t$ units to obtain the corresponding $\Delta(T^*_{p_j})^i$. The rotation operation visualizes the (14), and we mark $\Delta(T^*_{p_j})^i$ with dark green triangles. Typically, $s_i.len \geq 1$, the rotation operation will form rotated areas. We color these areas with gray shading and can efficiently find all new PriA time slots within them. Similarly, we construct other sub-mappings separately for the other periods that have different greatest common divisors with p_j. As a result, we can obtain all new PriA time slots contained in $\Delta(T^*_{p_j})^i$ from all these sub-mappings.

Overall, the proposed CoTiSCoM can visualize the computation of $\Delta(T^*_{p_j})^i$, as well as enable us to directly plot $\Delta(T^*_{p_j})^i$

Algorithm 1: Solver-Based Period-Aware Time Slot Allocation (PATSA-S) Strategy.

Input: S, P, R
Output: the Gate Control Lists (GCLs)
// network initialization
1: $T^*_{p_1}, T^*_{p_2}, \ldots, T^*_{p_n} \leftarrow \emptyset$
2: **for** $p_j \in P$ **do**
3: **for** $p_w \in P$ **do**
4: $\Delta'(T^*_{p_j})^{p_w} \leftarrow$ Equations (10)–(13)
5: **end for**
6: **end for**
$//timeslotallocationandT^*updating$
7: *ConstraintsSet* $\leftarrow$ S, R, scheduling constraints, optional constraint (15)
8: *Solver()* $\leftarrow$ *ConstraintsSet*
9: **if** *Solver()* is true **do**
10: GCLs $\leftarrow$ *Solver()*.results
11: **for** $s_i \in S$ **do**
12: **for** $[v_a, v_b] \in s_i.r$ **do**
13: **for** $p_j \in P$ **do**
14: $\Delta(T^*_{p_j})^i \leftarrow \Delta'(T^*_{p_j})^{s_i.p}$, $s_i^{[v_a,v_b]}.t \in$ GCLs, $s_i.len$, (14)
15: add $\Delta(T^*_{p_j})^i$ to $T^*_{p_j}$
16: **end for**
17: update T^* on $[v_a, v_b] \leftarrow T^*_{p_j}$
18: **end for**
19: **end for**
20: **else**
21: **return** unschedulable
22: **end if**

after the greatest common divisors between different periods are calculated.

V. Period-Aware Time Slot Allocation Strategy

After proposing the PriA time slots and designing its computational process, we propose two optimized strategies for corresponding scheduling algorithm types. Specifically, we propose a solver-based period-aware time slot allocation (PATSA-S) strategy for solver-based scheduling algorithms and a heuristic period-aware time slot allocation (PATSA-H) strategy for heuristic scheduling algorithms. In order to guarantee conflict-free scheduling for TT traffic, the proposed period-aware time slot allocation strategies centrally allocate the specific time slots to each TT stream and follow the classic scheduling constraints from [24]. Besides, in online scheduling scenarios, if our proposed strategies cannot allocate conflict-free time slots to the newly arrived streams due to the limitations of computed schedules, the TT traffic should be rerouted or rescheduled. It is worth noting that the specific rerouting and rescheduling methods are beyond the scope of the time slot allocation process. The PATSA-S and PATSA-H strategies are described in detail below.

A. Solver-Based Period-Aware Time Slot Allocation Strategy

The PATSA-S strategy comprises three phases: initialization, time slot allocation, and PriA time slots updating. The specific process is described in Algorithm 1, where notation R denotes the set of paths for all TT streams.

Specifically, during the initialization phase, the PriA time slots for each period are first initialized to the empty set (line 1). After that, the baseline $\Delta'(T^*_{p_j})^{p_w}$ for each period $p_w \in P$ is computed according to (10)–(13) (lines 2-6).

Subsequently, the PATSA-S strategy constructs the set of scheduling constraints based on the set of TT streams and their paths R (lines 7-8). Additionally, we propose an optional time slot allocation constraint to optimize the schedulability of TT traffic. This constraint limits that each TT stream $s_i \in S$ is preferentially allocated the time slots from $T^*_{s_i.p}$ along its path $s_i.r$. The time slot allocation constraint is formulated as follows:

$$\forall s_i \in S,\ \forall [v_a, v_b] \in s_i.r,\ \forall k \in \left[1, |T^*_{s_i.p}|\right],\ \forall t^*_{s_i.p,k} \in T^*_{s_i.p}:$$
$$s_i^{[v_a,v_b]}.t = t^*_{s_i.p,1} \bigvee s_i^{[v_a,v_b]}.t = t^*_{s_i.p,2} \bigvee \cdots$$
$$\bigvee s_i^{[v_a,v_b]}.t = t^*_{s_i.p,|T^*_{s_i.p}|}, \tag{15}$$

where $|T^*_{s_i.p}|$ denotes the number of time slots contained in $T^*_{s_i.p}$. The time slot allocation constraint is only satisfied if it does not conflict with other constraints. Therefore, the proposed constraint does not expand the solution space, and can be easily integrated into existing scheduling solvers.

Furthermore, if a feasible solution is found within the solution space constructed by the scheduling constraints, the PATSA-S strategy updates the PriA time slots for different periods (lines 9-19). Otherwise, the strategy returns an unschedulable status (lines 20-21).

B. Heuristic Period-Aware Time Slot Allocation Strategy

Similar to the PATSA-S strategy, the PATSA-H strategy also adopts a three-phase structure. The specific process of the PATSA-H strategy is described in Algorithm 2. The main difference between PATSA-H and PATSA-S is that PATSA-H incrementally allocates time slots to each sorted TT stream and then updates T^*, instead of allocating time slots to all TT streams at once.

Specifically, on each link in the transmission path of each sorted TT stream s_i, the PATSA-H strategy prioritizes the allocation of time slots contained in $T^*_{s_i.p}$ for s_i and removes the considered time slots from $T^*_{s_i.p}$ (lines 8-15). If these time slots cannot transmit s_i without conflicts with the scheduled TT streams, PATSA-H allocates time slots to s_i according to the as soon as possible allocation strategy (ASAP) (lines 16-19). After allocating time slots to s_i, PATSA-H updates the PriA time slots on link $[v_a, v_b]$ (lines 20-24).

In order to ensure the timely transmission of TT streams within their respective deadlines, the PATSA-H strategy filters out the larger time slots when adding new time slots to $T^*_{p_j}$, i.e., it modifies the (6) as follows:

$$T^*_{p_j} = T^*_{p_j} \cup f(\Delta(T^*_{p_j})^i). \tag{16}$$

Algorithm 2: Heuristic Period-Aware Time Slot Allocation (PATSA-H) Strategy.

Input: G, S, P, R
Output: the Gate Control Lists (GCLs)
// network initialization
$T^*_{p_1}, T^*_{p_2}, \ldots, T^*_{p_n} \leftarrow \emptyset$
for $p_j \in P$ **do**
  for $p_w \in P$ **do**
    $\Delta'(T^*_{p_j})^{p_w} \leftarrow$ Equations (10)–(13)
  end for
end for
//time slot allocation and T^* updating
sorting TT streams
for each sorted $s_i \in S$ **do**
  for $[v_a, v_b] \in s_i.r$ **do**
    while $T^*_{s_i.p}$ on link $[v_a, v_b]$ is not empty **do**
      remove the first element $t^*_{s_i.p}$ from $T^*_{s_i.p}$
      if $t^*_{s_i.p} = s_i^{[v_a,v_b]}.t$ is valid **then**
        GCLs $\leftarrow (s_i^{[v_a,v_b]}.t = t^*_{s_i.p})$
      end if
    end while
    if s_j has not been allocated time slots **do**
      $s_i^{[v_a,v_b]}.t \leftarrow$ the ASAP strategy
      GCLs $\leftarrow s_i^{[v_a,v_b]}.t$
    end if
    for $p_j \in P$ **do**
      $\Delta(T^*_{p_j})^i \leftarrow \Delta'(T^*_{p_j})^{s_i.p}$, $s_i^{[v_a,v_b]}.t \in$ GCLs, $s_i.len$, G, Equations (14), (16)–(17)
      add $\Delta(T^*_{p_j})^i$ to $T^*_{p_j}$
    end for
    update T^* on $[v_a, v_b] \leftarrow T^*_{p_j}$
  end for
end for

The filtering method can be adjusted according to different scenarios, and the default method in the PATSA-H strategy is as follows:

$$f(Z) = \{z \mid z \leq s_i.d - \gamma * r_{\max} * maxlen, z \in Z\}, \quad (17)$$

where $\gamma * r_{\max} * maxlen$ is the reserved time slots ensuring s_i traverses its path $s_i.r$, γ is a custom parameter, and $r_{\max}$ denotes the maximum path length between any two devices in the TSN topology G.

C. Adaptive Scheduling Scenarios

Depending on whether the transmission requirements of TT traffic change dynamically, scheduling scenarios in TSN can be classified as offline or online. Because constraint (15) contained in the PATSA-S strategy is optional, this strategy does not expand the solution space, but efficiently exploits it and guides the time slot allocation process. Therefore, the PATSA-S strategy can optimize the schedulability of newly added TT streams, making it particularly promising for the increasingly prevalent online scheduling scenarios. In addition, since the other proposed PATSA-H strategy allocates time slots incrementally, it has good generality and can be employed in both offline and online scheduling scenarios.

Furthermore, our proposed strategies are well-suited to scenarios where low-priority non-periodic traffic and best-effort (BE) traffic arrive randomly. Specifically, based on the IEEE 802.1Qbv standard, low-priority traffic can coexist with scheduled TT traffic by being transmitted within the transmission intervals between TT streams. Since the PriA time slots for different periods are typically dispersed within their corresponding periods, the randomly arriving non-periodic traffic and BE traffic can be served promptly.

VI. Simulation Results and Analysis

To comprehensively evaluate the optimization effects of the PATSA-S and PATSA-H strategies on schedulability, we compare them with the mainstream strategies in offline scheduling, incremental online scheduling, and reschedulable online scheduling scenarios. The difference between the two types of online scheduling scenarios is whether the TT traffic is rescheduled if the newly added TT streams cannot be allocated conflict-free transmission time slots. In this section, the simulation results are presented and analyzed in detail.

A. Simulation Setup

Since the network topology and the properties of TT traffic are the most significant factors affecting schedulability, they are the primary parameters considered in our simulation experiments. Specifically, as shown in Fig. 6(a) and (b) correspondingly, we set up the realistic Orion Crew Exploration Vehicle (CEV) network topology [31] and a classical in-vehicle network topology [32] as the topologies for the simulations. In addition, the bandwidth for TT traffic on all links is set to 1 Gbps, and each egress port of each TSN switch is allocated five queues for transmitting TT traffic.

For the properties of TT streams, we set the senders and receivers of each TT stream s_i to be random, the data size of s_i to belong to {100 B, 200 B, 300 B,..., 1500 B}, and route s_i according to the shortest path routing. In addition, we set the parameter γ in (17) equal to 1 for offline scheduling and 4 for online scheduling. All simulations are performed on Python 3.7 and Intel i5-7500 CPU with 3.4 GHz.

Furthermore, based on the existing studies represented by [33] and [34], we neglect the processing and propagation delays in the simulations. Specifically, Bosk et al. [33] and Liu et al. [34] have compared simulations with hardware measurements and found that the processing delay in hardware TSN switches is less than a few microseconds, while the propagation delay in short-haul cables is only a few nanoseconds. In comparison, the traffic periods are on the order of hundreds to thousands of microseconds in our simulation experiments. Besides, based on the IEEE 802.1Qbv standard, the end-to-end latency of a single packet is typically on the order of tens of microseconds with zero-jitter under ideal conditions, and potentially hundreds of microseconds if

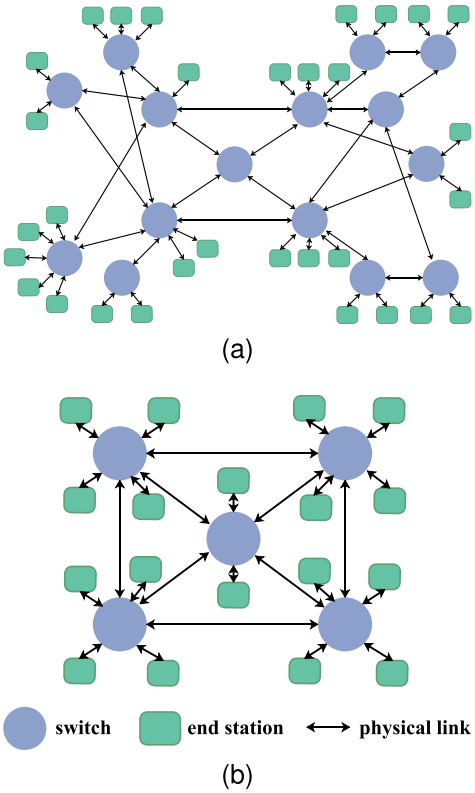

Fig. 6. Network topologies in the simulation experiments. (a) The realistic orion CEV network topology. (b) A classical in-vehicle network topology.

TABLE II
THE SETTING OF TRAFFIC PERIODS IN OFFLINE SCHEDULING

ID	Topology	Periods (μs)	Hyper-period(μs)
Group A	Orion CEV	$\{200, 500, 800, 1000, 1500\}$	12000
Group B	Orion CEV	$\{200, 400, 800, 1600, 3200\}$	3200
Group C	In-vehicle	$\{200, 500, 800, 1000, 1500\}$	12000
Group D	In-vehicle	$\{200, 400, 800, 1600, 3200\}$	3200

additional queuing is required. More importantly, the processing delay in each TSN switch and the propagation delay in each physical link are typically fixed values in the same network and will have the same impact on the transmission of different TT streams. As a result, the processing delay and the propagation delay in TSN are negligible in the simulation experiments.

B. Offline Scheduling

In offline scheduling scenarios, we fully evaluate the schedulability and runtime of the proposed PATSA-H strategy. As shown in Table II, we set four groups of simulation experiments with different topologies and period sets. The period set {200,500,800,1000,1500} features randomly spaced periods, while {200,400,800,1600,3200} features multiplicative growth, and the deadline for each TT stream is equal to its respective period. In addition, we compare the proposed PATSA-H strategy with the mainstream heuristic time slot allocation strategies: ASAP [26], ALAP [25], and No-Wait-H [28]. Furthermore, we

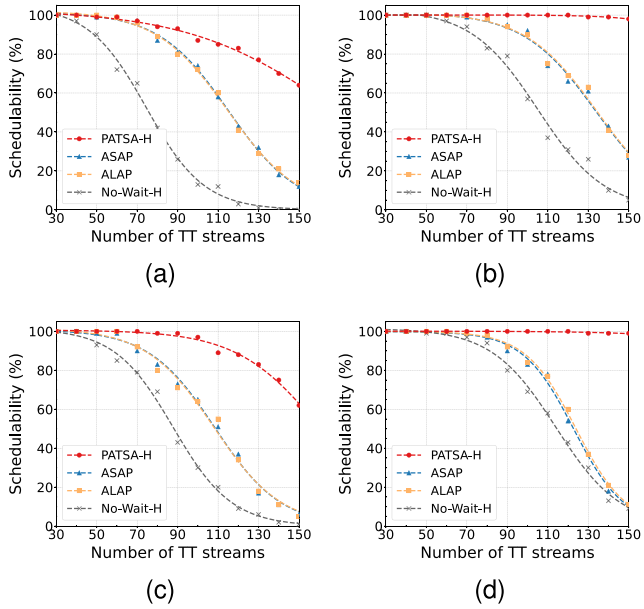

Fig. 7. The schedulability of different time slot allocation strategies in different offline simulation groups. (a) In group A. (b) In group B. (c) In group C. (d) In group D.

set the number of TT streams to range from 30 to 150, with intervals of 10 streams. For each setting, we run the simulation experiments 100 times, and calculate the schedulability of TT traffic and the average runtime of each strategy as the simulation results.

Fig. 7 shows the schedulability of different time slot allocation strategies across different simulation groups. In addition, the smooth curves in Fig. 7 fit the discrete simulation points of the corresponding strategies, which reflect the trend of schedulability with the growth of the TT traffic scale. The results demonstrate that the proposed PATSA-H strategy can schedule more TT streams under different topologies and period sets, particularly as the number of TT streams increases. For example, when scheduling 150 streams offline, the PATSA-H strategy can schedule more than 60% of the scheduling instances across the different simulation groups, while the schedulability of all comparison strategies is below 30%. In addition, the PATSA-H strategy generates more PriA time slots when the traffic periods increase multiplicatively. Therefore, as shown in Fig. 7(b) and (d), the PATSA-H strategy significantly improves the schedulability of the TT traffic in this case compared to other strategies. Furthermore, the fitted curves in Fig. 7 show that the ASAP and ALAP strategies have similar schedulability due to the periodic nature of TT streams. Moreover, the No-Wait-H strategy suffers from the worst schedulability in offline scheduling simulations, primarily due to its requirement for immediate forwarding of TT streams at switches, which reduces the solution space.

Fig. 8 presents the average runtime of each evaluated time slot allocation strategy and its fitted curve for the corresponding discrete simulation points. Besides, the colored area surrounding each curve represents the 95% confidence interval of the average runtime results under 100 repetitions of each simulation

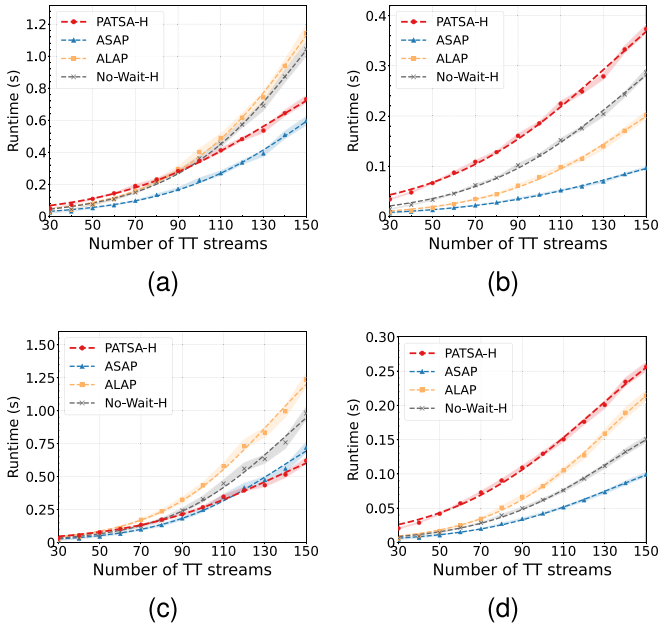

Fig. 8. Average runtime of different time slot allocation strategies across different offline simulation groups, with the colored area representing the 95% confidence interval of the average runtime results. (a) Group A. (b) Group B. (c) Group C. (d) Group D.

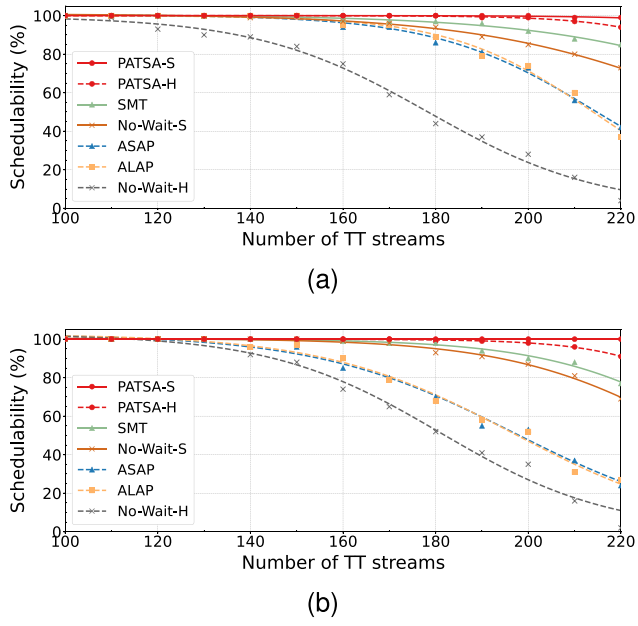

Fig. 9. In incremental online scheduling simulation experiments, the schedulability of different time slot allocation strategies in different topologies. (a) In orion CEV topology. (b) In in-vehicle topology.

scenario. The results show that the width of the 95% confidence intervals in all simulation experiments is less than 20% of the corresponding sample means, indicating low variation in runtime across different offline scheduling scenarios. In addition, the runtime of the PATSA-H strategy is consistently below 1 s in all simulations, although computing PriA time slots introduces additional overhead. More importantly, the PATSA-H strategy considers the PriA time slots with priority, which can help prevent the complex searches for allocation. Therefore, the runtime disadvantage of the PATSA-H strategy gradually decreases or even reverses as the number of TT streams increases. For example, as shown in Fig. 8(c), the PATSA-H strategy allocates the time slots to TT traffic faster than all comparison strategies when the number of TT streams is greater than 120, demonstrating its scalability and potential for application in large-scale TSN.

C. Incremental Online Scheduling

In online scheduling scenarios with dynamically increasing TT traffic, the scheduling space will gradually decrease. When the previous scheduling solution cannot accommodate the newly arrived TT streams, a common method is to reschedule the scheduled TT streams. However, the rescheduling of TT traffic usually introduces significant runtime overhead, which is preferably avoided in real-time scenarios. In this section, we compare the proposed PATSA-S and PATSA-H strategies with other mainstream allocation strategies in the incremental online scheduling scenarios without rescheduling.

For solver-based time slot allocation strategies, we utilize OpenPlanner [35], an open-source scheduling framework, to allocate the time slots to TT streams based on the SMT solver. The comparison solver-based strategies include the strategy based on the SMT solver [24] and the strategy that further employs the no-wait rule [27], which are labeled as SMT and No-Wait-S, respectively. We also set the heuristic strategies ASAP [26], ALAP [25], and No-Wait-H strategy [28] as comparison strategies.

In addition, the period of each TT stream in the incremental online scheduling is randomly selected from 300 μs, 600 μs, 900 μs, and 1200 μs, with probabilities of 0.125, 0.125, 0.25, and 0.5, respectively. To ensure statistical robustness, we vary the number of TT streams from 100 to 220, with intervals of 10 streams, and run the simulation experiments 100 times under each stream number setting. Moreover, the experiments schedule 50 TT streams offline at once, followed by dynamic scheduling of newly arrived TT streams in batches of 10.

The simulation results shown in Fig. 9 demonstrate the schedulability optimization of the proposed strategies in different topologies. Because the PATSA-S strategy can guide the solver to efficiently exploit the entire solution space based on the traffic periods and their allocated residue classes, it successfully schedules all TT streams under different simulation settings. In addition, the fitted curves in Fig. 9 show that SMT and No-Wait-S, which globally search for feasible solutions, have higher schedulability than the heuristic ASAP, ALAP, and No-Wait-H strategies. However, SMT and No-Wait-S compute arbitrary feasible solutions based on the current scheduling constraints during each online phase, potentially degrading the solution space of the newly arrived TT streams. In contrast, our proposed PATSA-H strategy dynamically computes PriA time slots and optimizes the solution space. Consequently, as shown in Fig. 9, its schedulability is consistently above 90%, only lower than the proposed PATSA-S strategy and higher than all the comparison strategies.

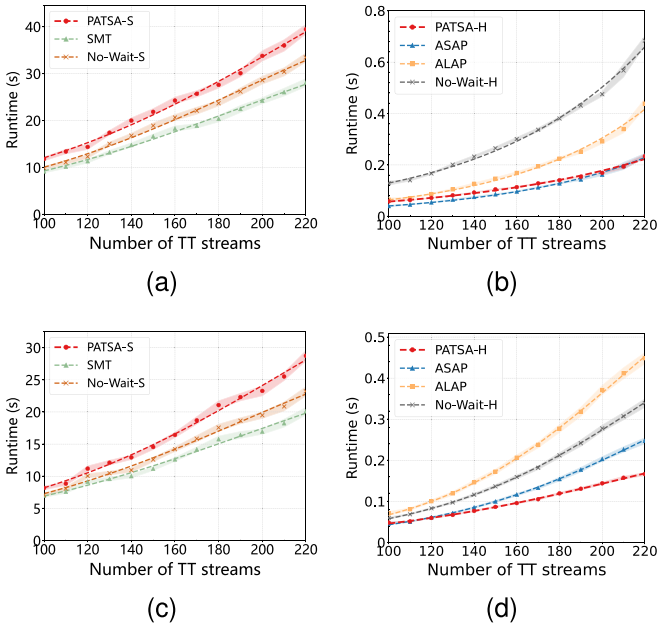

Fig. 10. In incremental online scheduling simulation experiments, the average runtime of different time slot allocation strategies in different topologies, with the colored area representing the 95% confidence interval of the average runtime results. (a) Solver-based strategies in orion CEV topology. (b) Heuristic strategies in orion CEV topology. (c) Solver-based strategies in in-vehicle topology. (d) Heuristic strategies in in-vehicle topology.

We also compute the average runtime of different time slot allocation strategies under 100 repetitions of each simulation topology, and mark the colored area in Fig. 10 to represent the 95% confidence interval of the corresponding simulation results. The width of the 95% confidence intervals in all simulation experiments is less than 15% of the corresponding sample means. As shown in Fig. 10(a) and (c), the runtime of the PATSA-S strategy remains in the same order of magnitude compared to the solver-based strategies, despite the additional runtime overhead caused by the optional slot allocation constraint. Considering the schedulability improvements achieved by the PATSA-S strategy, this additional runtime overhead is acceptable. Furthermore, similar to the offline scheduling results, the fitted curves in Fig. 10(b) and (d) show that the runtime of the PATSA-H strategy in online scheduling is close to or even shorter than that of the heuristic comparison strategies.

By comprehensively comparing the simulation results of the PATSA-S strategy and the PATSA-H strategy shown in Figs. 9 and 10, we can find that the PATSA-S strategy requires greater runtime overhead while achieving higher schedulability. These results indicate that the selection of our proposed strategies should be based on the specific requirements in terms of schedulability and scheduling runtime.

D. Reschedulable Online Scheduling

After demonstrating the effectiveness of the PATSA-S strategy and the PATSA-H strategy in incremental online scheduling, we further evaluate the two proposed strategies in reschedulable online scheduling. In large-scale dynamic scenarios, because the static nature of the precomputed schedules makes it challenging to accommodate the dynamic TT traffic, the rescheduling of the scheduled TT traffic is usually required.

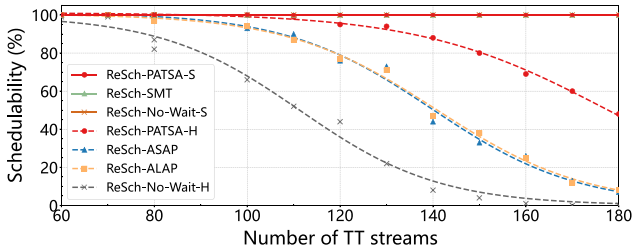

Fig. 11. In reschedulable online scheduling simulation experiments, the schedulability of different time slot allocation strategies in the orion CEV topology.

The simulation experiments in this section build upon the classic reconfiguration framework presented in [18], with a specific focus on optimizing the time slot allocation process and recomputing the scheduling solutions. Additionally, since the reconfiguration approach proposed in [18] does not specify the time slot allocation strategy, we separately integrate it with the proposed period-aware strategies and other existing time slot allocation strategies.

During the rescheduling process, the simulation experiments will recompute the scheduling for the scheduled streams and the newly arrived streams. In particular, the heuristic time slot allocation strategies sort the known TT streams in ascending order of data size before recomputation. If some streams have the same data size, they are sorted in ascending order of their period values. In order to fully utilize the network resources and prevent scheduling conflicts between TT streams, the simulation experiments compute the scheduling solutions in a centralized manner.

The period of each TT stream in the reschedulable online scheduling is randomly selected from 200 μs, 400 μs, 600 μs, and 800 μs, with probabilities of 0.125, 0.125, 0.25, and 0.5, respectively. This period setting restricts the transmission of TT traffic to more stringent deadlines, allowing an efficient evaluation of how different time slot allocation strategies affect the recomputation of the scheduling solution. In addition, the reschedulable online scheduling simulations are conducted in the realistic Orion CEV topology, and the number of TT streams is set from 60 to 180, with intervals of 10 streams. The rest of the experimental setup is the same as the incremental online scheduling experiments without rescheduling.

Fig. 11 presents the schedulability of different time slot allocation strategies in reschedulable online scheduling experiments. The results show that all the solver-based time slot allocation strategies can allocate time slots to all scheduling instances of TT traffic, indicating that the rescheduling method effectively exploits the advantage of the SMT solver to explore the entire solution space.

On the contrary, limited by the partial exploration of the solution space, the schedulability of all heuristic strategies in the experiments gradually decreases as the TT traffic scale increases. However, the fitted curves in Fig. 11 show that the schedulability of the PATSA-H strategy applying PriA time slots is significantly

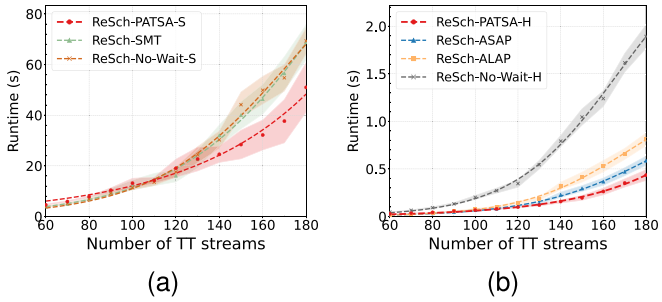

Fig. 12. In reschedulable online scheduling simulation experiments, the average runtime of different time slot allocation strategies in the orion CEV topology, with the colored area representing the 95% confidence interval of the average runtime results. (a) Solver-based strategies. (b) Heuristic strategies.

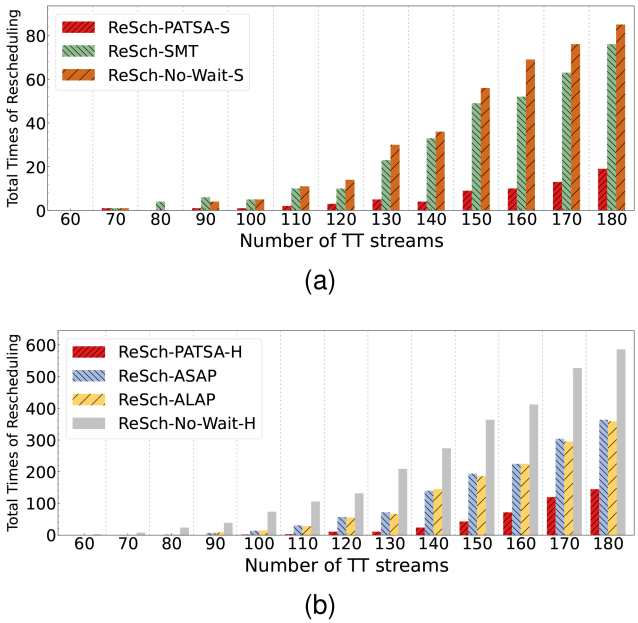

Fig. 13. In reschedulable online scheduling simulation experiments, the total times of rescheduling triggered by different time slot allocation strategies in the orion CEV topology. (a) Solver-based strategies. (b) Heuristic strategies.

higher than that of other comparison heuristic time slot allocation strategies, which verifies the effectiveness of the proposed PATSA-H strategy in reschedulable online scheduling scenarios.

In addition, Fig. 12 presents the average runtime of different time slot allocation strategies in the Orion CEV topology. When the number of TT streams is greater than 130, both our proposed PATSA-S and PATSA-H strategies have shorter average runtimes than the corresponding types of existing time slot allocation strategies. This result shows that although the computation of PriA time slots in our proposed period-aware strategies introduces additional runtime overhead, our proposed strategies have lower overall runtime overhead in the reschedulable online scheduling scenarios.

The colored area in Fig. 12 represents the 95% confidence interval of the average runtime results. Specifically, the width of the confidence intervals ranges from 6% to 35% of the sample means for the ReSch-SMT and ReSch-No-Wait-S strategies, from 7% to 46% for the ReSch-PATSA-S strategy, and less than 25% for the heuristic time slot allocation strategies. These confidence intervals are larger than those in Figs. 8 and 10, primarily because recomputing the scheduling solution requires extra time. In general, whether rescheduling is triggered substantially affects the runtime of each simulation, causing large time variations across the 100 repetitions of the experiment. In particular, the computation time of the SMT solver increases non-linearly with the number of TT streams scheduled simultaneously. Consequently, the runtime overhead for rescheduling all existing TT streams simultaneously is significantly greater than that for scheduling TT streams in batch. Additionally, the ReSch-PATSA-S strategy exhibits slightly wider confidence intervals than the ReSch-SMT and ReSch-No-Wait-S strategies, indicating that the rescheduling process under the PATSA-S strategy would cause larger time variations due to its extra time slot allocation constraint.

To analyze the reason why our proposed PATSA-S and PATSA-H strategies can have lower runtime overhead in reschedulable online scheduling, we further evaluate the total times of rescheduling triggered by different time slot allocation strategies within 100 repetitions of each TT traffic scale. The triggered rescheduling means that the corresponding time slot allocation strategy cannot incrementally schedule the TT streams and will introduce additional time overhead. As shown in Fig. 13, the total rescheduling times for all time slot allocation strategies increase with the TT traffic scale. In addition, the vertical coordinate range of Fig. 13(a) is 0-80, while that of Fig. 13(b) is 0-600, which indicates that the total rescheduling times triggered by the solver-based time slot allocation strategies are fewer than those triggered by the heuristic strategies. More importantly, the results show that the total times of rescheduling triggered by our proposed period-aware time slot allocation strategies are significantly fewer than those triggered by the corresponding types of time slot allocation strategies. Combined with Figs. 12 and 13, the experimental results demonstrate that our proposed strategies optimize the time slot allocation process and efficiently utilize the residue classes of the allocated time slots. As a result, our proposed strategies can significantly mitigate the runtime overhead and potential delays caused by frequent rescheduling of TT traffic in large-scale online scheduling.

Moreover, to evaluate how often the scheduling process needs to recompute the scheduling solutions, Fig. 14 further presents the distribution of TT flow rescheduling. The results show that under the experimental setup, the rescheduling distribution of each strategy is close to a normal distribution. Specifically, the number of TT streams during rescheduling is concentrated in 120-160 for the solver-based time slot allocation strategies, while those for the heuristic time slot allocation strategies are mainly concentrated in 100-150. In addition, the PATSA-S and PATSA-H strategies consistently exhibit lower rescheduling frequencies than their comparison strategies under different TT traffic scales.

Combined with Figs. 13 and 14, the experimental results show that the rescheduling frequency is closely related to the TT traffic scale and time slot allocation strategy. When the total number of TT streams ranges from 110 to 180, the solver-based time

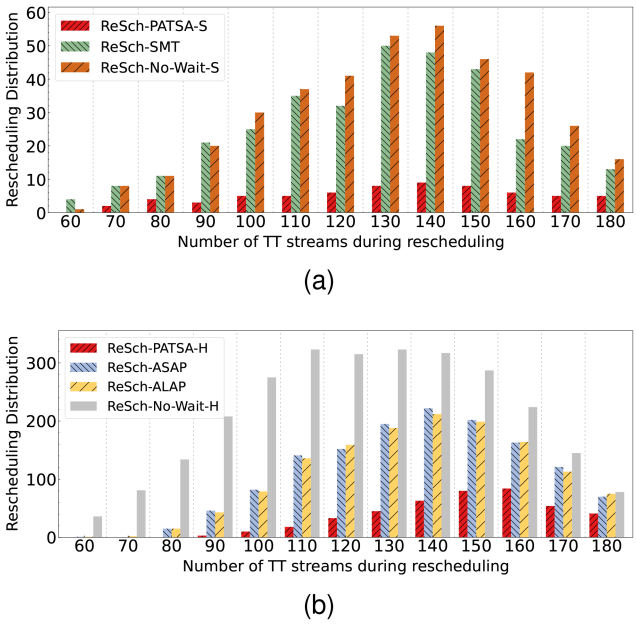

Fig. 14. In reschedulable online scheduling simulation experiments, the rescheduling distribution of different time slot allocation strategies in the orion CEV topology. (a) Solver-based strategies. (b) Heuristic strategies.

slot allocation strategies can typically accommodate the newly arrived TT streams by recomputing the scheduling solution at most once, while the heuristic time slot allocation strategies may require recomputing the scheduling solution several times. However, Fig. 11 shows that the schedulability of the heuristic strategies is still lower than that of the solver-based strategies.

In conclusion, the reschedulable online scheduling experiments further demonstrate that the PATSA-S and PATSA-H strategies have their own advantage in terms of schedulability performance and computational efficiency. More importantly, based on the schedulability analysis, these two proposed period-aware strategies can significantly improve the schedulability of the TT traffic and avoid the huge rescheduling overhead compared to the existing time slot allocation strategies. Considering the actual GCLs reconfiguration process in large-scale scenarios, our proposed strategies can further reduce the potential GCLs update delay and overhead.

VII. Conclusion and Future Work

In this paper, we have proposed a novel period-aware time slot allocation to optimize the schedulability of TT traffic in TSN, which complies with the IEEE 802.1Qbv standard and provides a new perspective on schedulability optimization. Based on abstract algebra, we analyze the impact of the time slot allocation strategy on schedulability and propose a concept called PriA time slots. In addition, we simplify the computational process of PriA time slots and propose CoTiSCoM to visualize it. Furthermore, according to the PriA time slots, we propose two time slot allocation strategies, PATSA-S and PATSA-H, to optimize the schedulability in corresponding scheduling algorithm types.

Extensive simulation experiments are conducted in offline scheduling, incremental online scheduling, and reschedulable online scheduling scenarios. The simulation results demonstrate that our proposed strategies achieve higher schedulability within acceptable runtime than existing time slot allocation strategies, which indicates their considerable potential in various real-time scenarios. In particular, under the reschedulable online scheduling, because our proposed strategies can avoid frequent rescheduling while optimizing schedulability, they have lower overall runtime overhead than that of the corresponding type of comparison strategies. Furthermore, comparing our proposed PATSA-S and PATSA-H strategies, the PATSA-S strategy can efficiently guide the time slot allocation process of the scheduling solvers and has the highest schedulability, while the PATSA-H strategy can allocate the TT streams within 1 s across the simulation groups.

In the future, we are interested in further studying the allocation order of different PriA time slots, and optimizing the schedulability under online scheduling without knowing the periods of the newly added TT streams in advance. These efforts are promising and will contribute to the further development of TSN in complex real-time scenarios.

Appendix A
Proof of Theorem 1

Without loss of generality, when allocating time slots to an arbitrary TT stream $s_i \in S$ on link $[v_a, v_b] \in E$ ($s_i.len = x$, and $x \in \{1, 2, \ldots, \text{maxlen}\}$), we assume that there are w residue classes modulo $s_i.p$ that are not allocatable residue classes and contains time slots allocated to the scheduled TT streams. In addition, we denote the minimum time slot in each of these w residue classes in ascending order as $a_1, a_2, a_3, \ldots, a_w$ ($a_w < s_i.p$), which divides $s_i.p$ into $(w+1)$ consecutive allocatable residue class segments. Then, we define L as an arbitrary segment among these segments. Specifically, $l_1 = a_1 - 0$, $l_2 = a_2 - a_1 - 1$, $l_3 = a_3 - a_2 - 1$,..., $l_w = a_w - a_{w-1} - 1$, $l_{w+1} = s_i.p - a_w - 1$, Therefore, L is a random variable representing the length of the chosen segment. Without loss of generality, we assume L follows a uniform distribution. Consequently, the expectation value of L is as follows:

$$exp(L) = \frac{a_1 - 0 + a_2 - a_1 - 1 + \cdots + s_i.p - a_w - 1}{w+1} = \frac{s_i.p - w}{w+1}. \tag{18}$$

According to Lemma 1, a necessary condition for s_i is schedulable on $[v_a, v_b]$ is $L \geq x$. Based on the Markov inequality, we have $P(L \geq x) \leq \frac{s_i.p-w}{x*(w+1)}$. Therefore, as the number of allocatable residue classes modulo $s_i.p$ decreases, the value of w increases, and the upper bound of $P(L \geq x)$ decreases. In other words, the upper bound of probability that s_i is schedulable on $[v_a, v_b]$ will also decrease.

Appendix B
Proof of Lemma 2

Because $hper$ is the least common multiple of all periods of given TT streams, it is not less than all those periods. Therefore, for the time slots within $hper$, the set of all residue classes modulo an arbitrary period $p_j \in P$ is $G_{p_j} = \{[0]_{p_j}, [1]_{p_j}, \ldots, [p_j - 1]_{p_j}\}$.

For any $a, b, c \in G_{p_j}$ and the addition operation, there is always $a + b \in G_{p_j}$ and $(a + b) + c = a + (b + c)$. Moreover, for $\forall a \in G_{p_j}$, there exists an identity element $[0]_{p_j}$ such that $a + [0]_{p_j} = a$, and there exists an inverse element $a^{-1} \in G_{p_j}$ such that $a + a^{-1} = [0]_{p_j}$. Therefore, G_{p_j} is a group.

Furthermore, there exists a generator $[1]_{p_j} \in G_{p_j}$, its exponents can form all the elements of G_{p_j}. In other words, $G_{p_j} = \{[1]^0_{p_j}, [1]^1_{p_j}, \ldots, [1]^{p_j-1}_{p_j}\}$, which means G_{p_j} is a cyclic group. As the number of elements in G_{p_j} is finite, G_{p_j} is a finite cyclic group.

APPENDIX C
PROOF OF THEOREM 2

First, we prove that for any $a, b \in G_{p_j,p_k}$, there is always $a + b \in G_{p_j,p_k}$. Without loss of generality, we set $a = [i_a * p_k \mod p_j]_{p_j}$, $b = [i_b * p_k \mod p_j]_{p_j}$. Therefore, $0 \le i_a, i_b \le \frac{hper}{p_k} - 1$ and $a + b = [(i_a + i_b) * p_k \mod p_j]_{p_j}$.

If $i_a + i_b < \frac{hper}{p_k}$, $a + b \in G_{p_j,p_k}$.

If $i_a + i_b = \frac{hper}{p_k}$, we have

$$\begin{aligned} a + b &= \left[\frac{hper}{p_k} * p_k \mod p_j\right]_{p_j} \\ &= [0]_{p_j}. \end{aligned} \tag{19}$$

Therefore, $a + b \in G_{p_j,p_k}$.

If $i_a + i_b > \frac{hper}{p_k}$, we have

$$\begin{aligned} a + b &= [(i_a + i_b) * p_k \mod p_j]_{p_j} \\ &= [(i_a + i_b) * p_k \mod p_j]_{p_j} - [0]_{p_j} \\ &= \left[\left(i_a + i_b - \frac{hper}{p_k}\right) * p_k \mod p_j\right]_{p_j}. \end{aligned} \tag{20}$$

Since $0 \le i_a, i_b \le \frac{hper}{p_k} - 1$, we have $\frac{hper}{p_k} + 1 \le i_a + i_b \le 2 * \frac{hper}{p_k} - 2$. Therefore, $1 \le (i_a + i_b - \frac{hper}{p_k}) \le \frac{hper}{p_k} - 2$. Consequently, $a + b \in G_{p_j,p_k}$.

In addition, for any $c = [i_c * p_k \mod p_j]_{p_j} \in G_{p_j,p_k}$. If $1 \le i_c \le \frac{hper}{p_k} - 1$, its inverse element is $c^{-1} = [(\frac{hper}{p_k} - i_c) * p_k \mod p_j]_{p_j}$. Because $1 \le \frac{hper}{p_k} - i_c \le \frac{hper}{p_k} - 1$, $c^{-1} \in G_{p_j,p_k}$. If $i_c = 0$, we also have $c^{-1} = [0]_{p_j} \in G_{p_j,p_k}$. Therefore, the inverse element of any element in G_{p_j,p_k} belongs to G_{p_j,p_k}.

Moreover, the identity element $[0]_{p_j}$ also belongs to G_{p_j,p_k}. Because G_{p_j,p_k} is a subset of group G_{p_j}, the associativity in G_{p_j} applies in G_{p_j,p_k} as well. In conclusion, G_{p_j,p_k} is a subgroup of G_{p_j}.

On this basis, G_{p_j,p_k} is also a finite cyclic group since G_{p_j} is a finite cyclic group. Furthermore, the order of $[p_k \mod p_j]_{p_j}$ is $\frac{p_j}{gcd(p_j,p_k)}$, where $gcd(p_j, p_k)$ denotes the greatest common divisor of p_j and p_k. Consequently, the number of elements in the finite cyclic group G_{p_j,p_k} is $\frac{p_j}{gcd(p_j,p_k)}$, and $[gcd(p_j, p_k)]_{p_j}$ is a generator of G_{p_j,p_k}.

REFERENCES

[1] T. Nie et al., "Hierarchical scheduling mechanism based on relaxed constraints for complex industrial scenes," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 3237–3249, May/Jun. 2024.

[2] "Time-sensitive networking task group," 2025. Accessed: May 12, 2025. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[3] L. Zhao, Y. Yan, and X. Zhou, "Minimum bandwidth reservation for CBS in TSN with real-time QoS guarantees," *IEEE Trans. Ind. Inform.*, vol. 20, no. 4, pp. 6187–6198, Apr. 2024.

[4] M. Li, S. Guo, C. Chen, C. Chen, X. Liao, and X. Guan, "DecAge: Decentralized flow scheduling for industrial 5G and TSN integrated networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 1, pp. 543–555, Jan./Feb. 2024.

[5] IEEE Time-Sensitive Networking Task Group, *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard 802.1Qbv-2015, 2016. Accessed: May 12, 2025. [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html

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

[7] H. Chahed and A. Kassler, "TSN network scheduling—-Challenges and approaches," *Network*, vol. 3, no. 4, pp. 585–624, Dec. 2023.

[8] L. Xu et al., "Learning-based scalable scheduling and routing co-design with stream similarity partitioning for time-sensitive networking," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13353–13363, Aug. 2022.

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

[10] D. Hellmanns, L. Haug, M. Hildebrand, F. Dürr, S. Kehrer, and R. Hummen, "How to optimize joint routing and scheduling models for TSN using integer linear programming," in *Proc. 29th Int. Conf. Real-Time Netw. Syst.*, 2021, pp. 100–111.

[11] S.-H. Chang, H. Chen, and B.-C. Cheng, "Time-predictable routing algorithm for time-sensitive networking: Schedulable guarantee of time-triggered streams," *Comput. Commun.*, vol. 172, pp. 183–195, Apr. 2021.

[12] J. Min, Y. Kim, M. Kim, J. Paek, and R. Govindan, "Reinforcement learning based routing for time-aware shaper scheduling in time-sensitive networks," *Comput. Netw.*, vol. 235, Nov. 2023, Art. no. 109983.

[13] A. Alnajim, S. Salehi, and C.-C. Shen, "Incremental path-selection and scheduling for time-sensitive networks," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2019, pp. 1–6.

[14] Y. Li, J. Jiang, and S. H. Hong, "Joint traffic routing and scheduling algorithm eliminating the nondeterministic interruption for TSN networks used in IIoT," *IEEE Int. Things J.*, vol. 9, no. 19, pp. 18663–18680, Oct. 2022.

[15] S. Yang, L. Zhuang, J. Lan, J. Zhang, and B. Li, "Reuse-based online joint routing and scheduling optimization mechanism in deterministic networks," *Comput. Netw.*, vol. 238, Jan. 2024, Art. no. 110117.

[16] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4525–4534, Jul. 2020.

[17] Z. Chen, Y. Lu, H. Wang, J. Qin, M. Wang, and W. Pan, "Dynamic stream partitioning for time-triggered traffic in time-sensitive networking," *Comput. Netw.*, vol. 248, Jun. 2024, Art. no. 110492.

[18] A. Nasrallah, V. Balasubramanian, A. Thyagaturu, M. Reisslein, and H. El-Bakoury, "Reconfiguration algorithms for high precision communications in time sensitive networks," in *Proc. 2019 IEEE Globecom Workshops*, Dec. 2019, pp. 1–6.

[19] N. S. Bulbul, D. Ergenc, and M. Fischer, "Towards SDN-based dynamic path reconfiguration for time sensitive networking," in *Proc. NOMS IEEE/IFIP Netw. Operations Manage. Symp.*, Apr. 2022, pp. 1–9.

[20] C. Gärtner, A. Rizk, B. Koldehofe, R. Guillaume, R. Kundel, and R. Steinmetz, "Fast incremental reconfiguration of dynamic time-sensitive networks at runtime," *Comput. Netw.*, vol. 224, Apr. 2023, Art. no. 109606.

[21] F. Kummer, F. Golatowski, W. Brekenfelder, H. Parzyjegla, P. Danielis, and G. Mühl, "An online scheduler for reconfigurable time-sensitive networks," in *Proc. IEEE 29th Int. Conf. Emerg. Technol. Factory Automat.*, Sep. 2024, pp. 1–8.

[22] M. Vlk, Z. Hanzálek, K. Brejchovvá, S. Tang, S. Bhattacharjee, and S. Fu, "Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1Qbv time-sensitive networks," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7023–7038, Nov. 2020.

[23] M. Vlk, K. Brejchovvá, Z. Hanzálek, and S. Tang, "Large-scale periodic scheduling in time-sensitive networks," *Comput. Operations Res.*, vol. 137, Jan. 2022, Art. no. 105512.

[24] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 183–192.
[25] D. Bujosa, M. Ashjaei, A. V. Papadopoulos, T. Nolte, and J. Proenza, "HERMES: Heuristic multi-queue scheduler for TSN time-triggered traffic with zero reception jitter capabilities," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, 2022, pp. 70–80.
[26] X. Zhou, F. He, L. Zhao, and E. Li, "Hybrid scheduling of tasks and messages for TSN-Based avionics systems," *IEEE Trans. Ind. Inform.*, vol. 20, no. 2, pp. 1081–1092, Feb. 2024.
[27] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 203–212.
[28] X. Jin, C. Xia, N. Guan, and P. Zeng, "Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 2, pp. 478–490, Feb. 2021.
[29] IEEE Time-Sensitive Networking Task Group, *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, IEEE Standard 802.1Qcc-2018, 2018. Accessed: May 12, 2025. [Online]. Available: https://1.ieee802.org/tsn/802-1qcc/
[30] IEEE Time-Sensitive Networking Task Group, *IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications*, IEEE Standard 802.1AS-2020, 2020. Accessed: May 12, 2025. [Online]. Available: https://standards.ieee.org/ieee/802.1AS/7121/
[31] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 2899–2928, Sep. 2022.
[32] B. Li, L. Zhuang, G. Wang, and Y. Sun, "Service-oriented data consistency research for in-vehicle Ethernet," *Veh. Commun.*, vol. 47, Jun. 2024, Art. no. 100776.
[33] M. Bosk et al., "Methodology and infrastructure for TSN-Based reproducible network experiments," *IEEE Access*, vol. 10, pp. 109203–109239, 2022.
[34] H.-H. Liu et al., "Improving TSN simulation accuracy in OMNeT++ : A hardware-aligned approach," *IEEE Access*, vol. 12, pp. 79937–79956, 2024.
[35] X. Jiang, "Opentsn/openplanner," 2022. Accessed: Oct. 22, 2024. [Online]. Available: https://gitee.com/opentsn/open-planner

Yiqin Lu (Member, IEEE) was born in Zhaoqin, Guangdong, China. He received the Ph.D. degree in electronic circuits and systems from the South China University of Technology (SCUT), Guangzhou, China, in 1996. He was appointed a Professor of SCUT in 2006. He is also the Doctoral Supervisor of the College of Electronic and Information in SCUT. He is currently the Dean of the office of the Leading Group for Cyberspace Affairs of SCUT, Director of the Network Center of Southern China Region of China Education and Research Network (CERNET), and President of Computer Information Network Safety Association of Guangdong Province. His research interests include communication network, network security, error-correcting code, and Internet of things.

Zhuoxing Chen was born in Guangzhou, Guangdong, China. He received the B.E. degree in communication engineering from Shandong University (SDU), Jinan, China. He is currently working toward the Ph.D. degree in information and communication engineering with the South China University of Technology (SCUT), Guangzhou, China. His research interests include deterministic communications, scheduling optimization, and cyberspace security.

Jiancheng Qin received the bachelor's degree in computer engineering from the School of Computer Science and Technology, Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1999, and the master's degree from the School of Software Engineering, BUPT, in 2008, and the Ph.D. degree from School of Computer, BUPT, in 2011. His major fields of study are information security and mobile computing.

Weiqiang Pan received the M.S. and Ph.D. degrees from the School of Electronics and Information Engineering, South China University of Technology, Guangzhou, China, in 2002 and 2007, respectively. In 1995, he joined the South China University of Technology, where he is currently a Professor with the School of Electronics and Information Engineering. His research interests include real-time systems, network architecture, and network security.