mathematics

MDPI

Article

Start Time Planning for Cyclic Queuing and Forwarding in Time-Sensitive Networks

Daqian Liu, Zhewei Zhang, Yuntao Shi *, Yingying Wang, Jingcheng Guo and Zhenwu Lei

School of Electrical and Control Engineering, North China University of Technology, Beijing 100144, China; liudaqian@ncut.edu.cn (D.L.); zzw159@mail.ncut.edu.cn (Z.Z.); wyy@ncut.edu.cn (Y.W.); guojingcheng@ncut.edu.cn (J.G.); leizhenwu@ncut.edu.cn (Z.L.)
* Correspondence: shiyuntao@ncut.edu.cn

Abstract: Time-sensitive networking (TSN) is a kind of network communication technology applied in fields such as industrial internet and intelligent transportation, capable of meeting the application requirements for precise time synchronization and low-latency deterministic forwarding. In TSN, cyclic queuing and forwarding (CQF) is a traffic shaping mechanism that has been extensively discussed in the recent literature, which allows the delay of time-triggered (TT) flow to be definite and easily calculable. In this paper, two algorithms are designed to tackle the start time planning issue with the CQF mechanism, namely the flow–path–offset joint scheduling (FPOJS) algorithm and congestion-aware scheduling algorithm, to improve the scheduling success ratio of TT flows. The FPOJS algorithm, which adopts a novel scheduling object—a combination of flow, path, and offset—implements scheduling in descending order of a well-designed priority that considers the resource capacity and resource requirements of ports. The congestion-aware scheduling algorithm identifies and optimizes congested ports during scheduling and substantially improves the scheduling success ratio by dynamically configuring port resources. The experimental results demonstrate that the FPOJS algorithm achieves a 39% improvement in the scheduling success ratio over the naive algorithm, 13% over the Tabu-ITP algorithm, and 10% over the MSS algorithm. Moreover, the algorithm exhibits a higher scheduling success ratio under large-scale TSN.

Keywords: industrial internet; time-sensitive networking; cyclic queuing and forwarding; joint routing scheduling; start time planning

MSC: 68T20; 68U07; 94A40; 49M05

check for updates

Citation: Liu, D.; Zhang, Z.; Shi, Y.; Wang, Y.; Guo, J.; Lei, Z. Start Time Planning for Cyclic Queuing and Forwarding in Time-Sensitive Networks. *Mathematics* **2024**, *12*, 3382. https://doi.org/10.3390/math12213382

Academic Editor: Alfredo Milani

Received: 2 October 2024
Revised: 26 October 2024
Accepted: 27 October 2024
Published: 29 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of real-time critical systems such as automobiles, aerospace, industrial automation, and industrial control, the ever-increasing communication data traffic puts forward high-bandwidth and low-delay requirements for communication networks [1]. Traditional Ethernet is limited to addressing non-real-time, best-effort data transmission, which poses challenges in fulfilling the diverse and hybrid traffic service transmission requirements demanded by the emerging industry for communication networks. While industrial networks like PROFINET, POWERLINK, and EtherCAT achieve different levels of real-time transmission performance, their compatibility and interoperability are limited by the proprietary nature of their mechanisms [2].

To address the aforementioned issues with Ethernet, TSN, a set of standards specified by IEEE 802 [3], emerged at a historic moment. The IEEE 802.1 TSN Task Group is actively working on the standardization of time-sensitive capabilities across 802 networks. This involves the development of a comprehensive suite of 802.1 sub-standards, which address various aspects including clock synchronization, frame preemption, redundancy management, and enhancements for scheduled traffic [4]. Specifically, the IEEE 802.1Qbv standard

outlines a gate mechanism known as the time-aware shaper (TAS), which regulates the selection of queues for connection to the corresponding egress port. This is achieved through a statically generated periodic schedule, referred to as the gate control list (GCL) [5]. GCL allows TT flow and unscheduled flows to be temporally isolated. There is an inherent complexity in generating the GCLs, i.e., finding the right points in time to open and close the gates [6], and demanding dynamic and complex configuration [7]. Consequently, the IEEE 802.1 Qch standard proposed an easy-to-use model named CQF. The principle underlying CQF is that TT flows are periodically queued and transmitted in TSN [8].

Even though CQF simplifies the gate control mechanism, constructing and optimizing data flow schedules to enhance scheduling performance remains a challenge [9]. Data flow scheduling strategies are an active field of research, often adopting a serialized scheduling algorithm. The general step of serialization scheduling is to first sort the flows based on a certain flow attribute (such as packet size, period, deadline, etc.). Then, the algorithm sequentially searches for the initial sending time (also called the offset) for each flow. Finally, the scheduling success is judged by the resource availability of each node through which the target flow passes. By precisely planning the sending times, scheduling failures caused by queue congestion can be effectively avoided, and scheduling performance can be improved. Most scheduling strategies tend to consider only the attributes of flow and do not adequately take into account the remaining capacity of all the port queues along the flow paths, which means that the resource allowance for the scheduling period is ignored. This often leads to congestion in port queues, which limits the flexibility of subsequent scheduling efforts. The resource margin of flow is determined by both path and offset. Only by establishing the path and offset of flow can the scheduling strategy calculate the resource consumption of the flow and determine the resource margin. The serialized scheduling cannot calculate flow offset until flow sorting is determined. Therefore, flow sorting, path planning, and offset search should not be isolated but should be considered comprehensively. To address the aforementioned issues, this study presents a FPOJS algorithm based on load balancing. The primary innovation of this algorithm lies in its comprehensive consideration of the interactions between flow sorting, path planning, and offset search. It integrates these three previously isolated steps into a unified combination, guided by a priority scoring mechanism, to achieve the objective of finding the optimal solution. Furthermore, the superiority of this method has been validated through experimental verification. The significant contributions of this paper can be summarized as follows.

1. The traditional incremental scheduling is improved by shifting from scheduling flow independently to scheduling a novel scheduling object: a combination of (*flow, path, offset*). The scheduling objects can integrate the independent scheduling processes of flow sorting, path planning, and offset search into a cohesive whole.
2. A flow–path–offset joint scheduling algorithm is proposed to schedule based on a novel multi-index priority scoring mechanism, which enhances the load balancing degree and increases the scheduling success ratio. The scoring mechanism depends on the total resources occupied by the scheduling object and the remaining port capacity. The scoring mechanism can accurately reflect the impact of each scheduling object on the scheduling process.
3. A congestion-aware scheduling algorithm is presented that can identify and optimize congested ports during scheduling. The reason for scheduling failure is not that the overall link load ratio is too high, but that the resources of a few ports along the path are congested. This method can detect congested ports early in the scheduling process and reduce scheduling failures caused by small amounts of port congestion.

The rest of this article is organized as follows. In Section 2, the related work is reviewed. Section 3 introduces the network model and CQF mechanism. The target optimization problem with CQF in TSN is presented in Section 4. Section 5 introduces the method for calculating the priority of combination and proposes the FPOJS algorithm. Section 6 presents the evaluation results. Finally, the conclusion is given in Section 7.

2. Related Work

In recent years, research on TSN has primarily encompassed time synchronization, traffic scheduling, reliability analysis, and routing planning. The study of TAS mainly focuses on the configuration of GCL and the planning of time slots. In early research, researchers attempted to address this NP-hard problem using solvers. Steiner et al. [10] proposed a strategy for solving the scheduling problem for time-triggered systems using SMT solvers and defined the formal scheduling constraints that apply in a time-triggered network. Drr et al. [11] modeled the traffic scheduling problem in TSN through the well-known no-wait job shop scheduling problem, proposing an integer linear programming (ILP) model along with a meta-heuristic optimization algorithm. These strategies ensured deterministic transmission while incurring a high time cost. Hence, based on the constraint framework proposed by [10], Nayak et al. [12] synthesized the GCL scheduling as a constrained system through first-order array theory and proved that this method had a universal SMT/OMT solver, which significantly reduced the required run time and improved the scalability. To meet the requirements in large-scale networks, Nayak et al. [13] designed a network architecture based on software-defined networking (SDN). It dynamically computed the routing and scheduling of TT flows through the logical centralization of the control plane, and on this basis, proposed multiple ILP formulations to compute incremental schedules [14] approximating the optimal solutions of their corresponding static scheduling problems.

As previously mentioned, CQF is a simplified model of TAS. Quan et al. [15], building upon the queue model and workflow of CQF, introduced the injection time planning (ITP) mechanism, which mapped TT flows onto the underlying resources both temporally and spatially. The research highlighted a critical issue within CQF, start time planning, and attempted to address it using a Tabu algorithm. This heuristic approach, while effective, incurred a high time cost and was not suitable for large-scale networks. Cheng et al. [16] proposed a deep reinforcement learning-assisted two-stage scheduling algorithm to solve the temporal scheduling problem based on the CQF model, which can be adopted in the off-line and on-line scenarios. Nie et al. [17] designed a method that adaptively adjusts the queue length and slot length according to the flow properties and proposed a start time planning method based on ILP. Zhang et al. [18] proposed a flow sequence analysis method based on divisibility theory after analyzing the mechanism of CQF and designed an incremental scheduling algorithm with flow judgment conditions and variable search boundaries. The aforementioned approaches represent various solutions to the NP-hard problem of initial send time planning within the ITP framework [15].

In the area of route planning, Chang et al. [19] designed an online scheduling framework and proposed a real-time routing scheduling algorithm based on end-to-end maximum delay, which aimed to assign priority according to the time period and determine the order of route processing by the algorithm. To address the issue of fault tolerance against permanent and transient faults for TT flows in TSN, Feng et al. [20] introduced a route scheduling algorithm based on redundant paths and offset allocation for online incremental rerouting and rescheduling of disrupted flows. Nie et al. [21] classified the periodicity of flows and designed an adaptive periodic compensation route scheduling algorithm, which improved the scheduling performance and reduced the execution time of the algorithm. Guo et al. [22] proposed a scheduling method that utilizes the interaction between flow and offsets. However, the research ignores the impact of path selection on overall scheduling. Wang et al. [23] proposed a load-balanced joint route scheduling algorithm based on the scheduling priority of each TT flow. Although this literature takes into account factors such as sorting, path selection, and offset search, it still adheres to the traditional approach of sorting first and then scheduling.

However, the above methods still adopt the serialized scheduling concept, which separates flow sorting, path planning, and offset search from each other, and proposes improvements from one aspect. In this paper, we design a parallel scheduling strategy that transforms the scheduling subject from a single flow to a combination of flow, path,

and offset. Meanwhile, a multi-index scoring mechanism based on the total amount of resources occupied by scheduling objects and the remaining port capacity is proposed. Moreover, based on the unique resource occupancy determinism of the scoring mechanism, the scheduling success ratio is enhanced through the optimization of congested resource blocks.

3. System Model

In this section, a comprehensive network model is built, which includes switches, topology, and data flow models. On this basis, a systematic CQF model is designed, which provides a detailed description of the basic principles and guidelines of CQF.

3.1. Network Model

The TSN network can be represented as a directed graph $N =< V, E >$, where V is the set of nodes which include terminal and switch nodes $v_i \in V$ and E is the set of directed links connecting nodes. The notation $(v_a, v_b) \in E$ denotes the actual physical link from node v_a to node v_b. In the model, each node is comprised of multiple ports, with each port encompassing eight priority queues ranging from Q_0 to Q_7. Assuming each queue has the same length, the two highest-priority queues, Q_6 and Q_7, are designated as the transmit and receive queues in the CQF mechanism [3]. The set of flows is denoted by F, with the attributes defined as follows:

$$f_k = \{f_k.src, f_k.dst, f_k.ddl, f_k.period, f_k.size, f_k.path, f_k.offset\} \quad (1)$$

where $f_k.src$ refers to the source node of the flow, $f_k.dst$ is the destination node of the flow, $f_k.ddl$ is the deadline of the flow, $f_k.period$ is the period of the flow, and $f_k.size$ is the size of the flow. It is assumed that each flow is considered to be a unicast flow, since multicast flows can be treated as a set of unicast flows. The path number of the flow f_k in scheduling is $f_k.path$, and $f_k.offset$ is the offset, or the send time when the first frame of flow f_k is transmitted from the source host. The set of shortest paths for a flow, represented by $path_j \in P$, is determined by the shortest path algorithm. The path is represented by the port numbers through which flow f_i is transmitted, typically denoted as a set of ports. The expression is as follows:

$$f_k.path = \{H_x, SW_i^m, SW_{i+1}^n, \cdots, H_y\} \quad (2)$$

where H_x and H_y represent the address of the source host and destination host, respectively. SW_i^m represents port m on node SW_i. Due to different flow periods and multiple repeats in the scheduling period, in order to avoid potential interference, the scheduling period is set to the hyper-period of the least common multiple of all flow periods. T_{sched} represents the hyper-period, which can be calculated based on the following:

$$T_{sched} = LCM(f_k.period). \quad (3)$$

The notations for this paper are listed in Table 1.

Table 1. List of main notations.

Parameters	Connotations
L_{slot}	Time slot length
T_{sched}	Hyper-period
SW_i	TSN switches

Table 1. *Cont.*

Parameters	Connotations
$f_k.offset$	Injection time slot of f_k
D_{proc}	Processing delay
D_{prop}	Propagation delay
BW	Link bandwidth
L_{Queue}	Queue length
h_k	Number of hops for flow f_k
$SQR(i,m,j)$	Slot queue resource of port m on node SW_i during $slot_j$
$O(f_k, SQR(i,m,j))$	Occupancy status of flow f_k at $SQR(i,m,j)$

3.2. CQF Scheduling Model

In the CQF model, the scheduling time is divided into equal time slots. The length of a slot is L_{slot}. The model is defined based on two principles:

1. The packet must be transmitted from the current node to the next node in the same slot.
2. Packets received by a node within $slot_k$ will be forwarded in $slot_{k+1}$.

Combining the above two principles and the goal of maximizing the number of schedulable flows, the following constraints can be derived. It is assumed that node SW_i transmits packets to its downflow neighbor node SW_{i+1}. In the worst case, if SW_j starts transmitting a packet to SW_{j+1} at the beginning of $slot_j$ and SW_{j+1} forwards it at the end of slot $slot_{j+1}$, i.e., transmission and reception occur at the farthest points of two adjacent slots, the delay between these two nodes is $2d$. In the best case, if SW_i transmits a packet to SW_{i+1} at the end of $slot_j$ and SW_{i+1} forwards it at the beginning of $slot_{j+1}$, i.e., transmission and reception occur at the adjacent ends of two slots, the delay between these two nodes is 0. In summary, assuming h is the number of hops, the maximum and minimum delays of a frame are as follows:

$$\begin{aligned} \text{maxDelay} &= (h+1)L_{slot} \\ \text{minDelay} &= (h-1)L_{slot} \end{aligned}. \tag{4}$$

Thus, it is evident that the end-to-end delay range of a frame is determined by the slot length and the number of hops, independent of the network topology. The node model that supports CQF is shown in Figure 1, where Q_6 and Q_7 are responsible for receiving and forwarding TT flows from each port. In a slot, one queue is in receive mode, while the other is in send mode, and the mode of each queue flips between adjacent slots to avoid additional computation and configuration overhead.

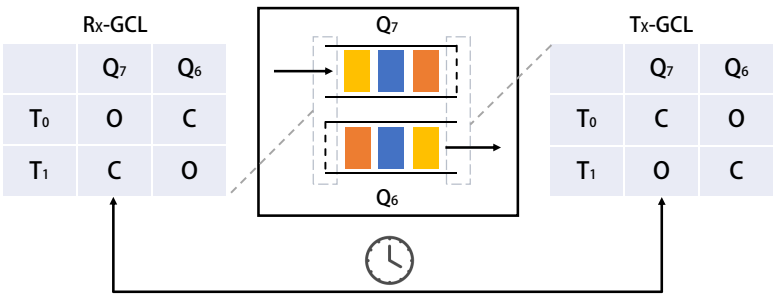


Figure 1. Framework of the CQF mechanism.

4. Problem Statement

In the CQF model, the transmission time planning of flow is the focus of this research. When scheduling resources are improperly allocated, TT flows tend to accumulate in queues at specific slots, which will cause TSN congestion. Therefore, a slot offset method that allocates initial sending time for each flow to avoid congestion is introduced. This method needs to meet the following constraints.

4.1. Slot Length Constraint

According to principle (2) mentioned in Section 3 , the transfer of all data flows in the queue between two adjacent nodes must be completed within a slot. Therefore, the lower bound of the slot length is constrained by the following:

$$\text{Min}(L_{slot}) = \frac{L_{Queue}}{BW} + D_{proc} + D_{prop} + D_{sync} \tag{5}$$

where L_{Queue} denotes the queue length; that is, the maximum byte capacity of each queue, BW represents the bandwidth of the link, D_{proc} and D_{prop} represent the internal processing delay and propagation delay, respectively, and D_{sync} refers to the delay induced by timing synchronization errors. To ensure that the repetition count of all flows within the scheduling period is an integer, the greatest common divisor (GCD) of the periods of all flows is utilized as the scheduling period. The upper bound on the slot length is defined as follows:

$$\text{Max}(L_{slot}) = GCD(f_k.period). \tag{6}$$

In the scheduling process, the slot duration is typically set as a constant value in accordance with the requirements of TSN.

In the CQF model, the arrival time of the flows depends solely on the number of hops and the slot offset. To ensure the QoS of TSN, flows must arrive before the deadline. The constraint on the offset is as follows:

$$f_k.offset + h_k + 1 \leq \frac{f_k.deadline}{L_{slot}} \tag{7}$$

where h_k represents the number of hops for flow f_k. Additionally, in the model of data flow, every flow is associated with a deadline. Delays in initiating a flow may result in failure to reach its intended destination by the deadline. To prevent such occurrences, a constraint is applied to guarantee that all flows arrive punctually, adhering to their specified deadlines. The offset constraint stipulates that the starting time of a data flow must fall within its designated period. Failure to comply with this constraint can result in the accumulation of multiple frames of the same flow in the host, thereby consuming valuable and limited buffer space. The constrains can be represented as follows:

$$f_k.offset < \frac{f_k.period}{L_{slot}}. \tag{8}$$

In addition, a single flow will have multiple frames, which requires a significant amount of terminal storage capacity.

4.2. Queue Resource Constraints

Each port of a node has fixed Slot Queue Resources (SQR) for each slot, where the size of SQR equals the length of the port queue L_{Queue}. $SQR(i,m,j)$ denotes the slot queue resource of port m on node SW_i during $slot_j$. A binary decision variable $O(f_k, SQR(i,m,j))$ represents the occupancy status of flow f_k at $SQR(i,m,j)$, and based on this occupancy, queue resource constraints are proposed as follows:

$$\begin{aligned}
&\forall f_k \in F, k \in [0, n-1], i \in [0, x-1] \\
&\forall m \in [0, SW_i.P-1], j \in [0, \frac{T_{sched}}{L_{slot}} - 1] \\
&O(f_k, SQR(i,m,j)) = \begin{cases} 0, SW_i^m \neq f_k.path[j] \\ 1, SW_i^m = f_k.path[j] \end{cases}
\end{aligned} \tag{9}$$

$$\sum_{k=0}^{n-1} O(f_k, SQR(i,m,j)) \times f_k.size \leq L_{Queue} \tag{10}$$

where n is the total number of flows, x is the number of nodes, and $SW_i.P$ is the total number of ports of node SW_i. When $O(f_k, SQR(i,m,j))$ is equal to 1, it indicates that flow f_k occupies the $SQR(i,m,j)$; otherwise, it is 0. Formula (9) represents the occupancy status of various flows within the system. Specifically, it indicates whether each flow is currently occupying buffer space or not. On the other hand, the queue resource constraint of a port is represented by Formula (10). This constraint stipulates that the total frame length of all frames currently stored in the Shared Queue Region (SQR) of the port cannot exceed the maximum queue length that the port can accommodate. In this article, the optimization objective is maximizing the number of successfully scheduled time-sensitive flows. This involves ensuring that each time-sensitive flow is allocated sufficient SQR resources to meet its transmission requirements within the given scheduling period.

5. Algorithm Design

As previously mentioned, the scheduling priority of flow is influenced by both the resource margin and requirements. The load balancing degree can be improved by transitioning the scheduling object from independent flow to a combination of flow, path, and offset . The multi-index priority scoring mechanism based on resource margin and requirements is introduced using a comprehensive example. The advantages of the novel scheduling object and scoring mechanism over the fixed order and fixed path methods are analyzed. The FPOJS algorithm based on the combinatorial scheduling object and multi-index priority scoring mechanism is proposed. Furthermore, the congestion-aware scheduling algorithm is proposed by calculating resource occupancy in real time during scheduling.

5.1. Scheduling Object and Priority Sorting Mechanism

The novel scheduling object, a combination of flow, path, and offset, and the multi-index priority sorting mechanism are the core of the FPOJS algorithm. In this section, the calculation method of the multi-index priority sorting mechanism is proposed, and the advantages of combinatorial scheduling objects are illustrated using an example.

The TSN topology and port information are shown in Figure 2. It is assumed that this topology is part of a TSN scenario. The attributes of flow are shown in Table 2. The resource requirements that affect the calculated load include packet size, hop count, and period. The total SQR occupied by a flow in a scheduling period can be calculated as follows:

$$O_{Total} = f_k.size \times f_k.hop \times \frac{T_{sched}}{f_k.period}. \tag{11}$$

The greater the resource requirements, the greater the impact on subsequent scheduling.

Assuming that all ports have an initial capacity of 100 and the remaining capacity of ports 2, 3, and 4 at each time slot remains at 100, the remaining capacity of the other ports is shown in Table 3. The numbers in the table indicate the remaining capacity of the ports for the corresponding time slots.

Given that the scheduling object is a combination of flow and offset, which occupies a specific amount of SRQs, congestion of any SRQ along the path can cause scheduling failures. Therefore, the resource margin is defined as the minimum remaining capacity of all SRQs occupied along the path. In Section 4, if flow f_k occupies $SQR(i,m,j)$, then $O(f_k, SQR(i,m,j))$ is equal to 1; otherwise, it is 0. Here, a set of flows $F_{SQR(i,m,j)} = \{f \mid O(f, SQR(i,m,j)) = 1\}$ represent all the flows that occupy the same SQR. The remaining capacity of the SQR is calculated as follows:

$$R_{SQR(i,m,j)} = L_{queue} - \sum_{f \in F_{SQR(i,m,j)}} f.size. \tag{12}$$

Then, the resource capacity is represented as $(MinR_{SQR(i,m,j)} - f_k.size)$.

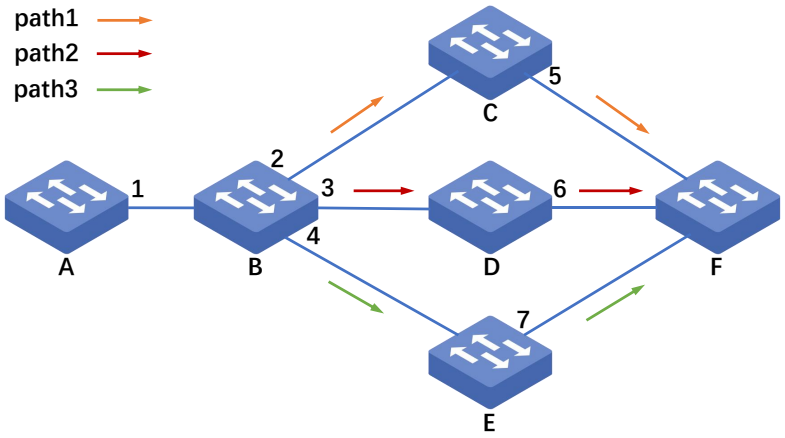


Figure 2. Port resources.

Table 2. Resource requirements.

Flow	Size	Route	Period	O_{total}
f_1	30	B→F	3	120
f_2	40	B→F	3	160
f_3	50	B→F	2	300
f_4	10	A→F	3	90

Table 3. Port capacity.

	Slot0	Slot1	Slot2	Slot3	Slot4	Slot5
Port1	15	15	15	15	15	15
Port5	80	60	50	80	60	50
Port6	40	30	70	40	30	70
Port7	50	90	40	50	90	40

In this example, when selecting path 1 for any flow in Table 1, it must pass through port 5. At this time, the resource margins of port 5 for selecting slot 0, slot 1, slot 2, slot 3, slot 4, and slot 5 are 80, 60, 50, 80, 60, and 50, respectively. As for f_4, since the remaining capacity of port 1 is 15 at all three time slots, which is less than the remaining capacity of any other port at any three time slots, the initial resource margin of f_4, regardless of the chosen path and time slot, is fixed at 15.

The scheduling success ratio is improved by reasonable planning of the link load; that is, each round of scheduling selects the scheduling object that has the lowest impact on the link load. Therefore, scheduling objects with high resource margins and low resource requirements should be prioritized. The formula for calculating the priority of scheduling objects through weight allocation is as follows:

$$Pr = \frac{\alpha \times (MinR_{SQR(i,m,j)} - f_k.size)}{\beta \times O_{Total}}. \tag{13}$$

The weight factors α and β can be adjusted according to the importance of the resource margin and requirements in actual applications. In this example, both α and β are equal to 1. The higher the value of Pr, the smaller the impact on the overall scheduling; hence, the scheduling object with the highest Pr is always chosen in scheduling.

In the first scheduling, the priority Pr of each combination is calculated, among which the combination $(f_4, path3, slot2)$ has a resource demand of 160 and a resource margin of 50; thus, $Pr = 50/160$. After calculation, this combination has the highest Pr among all

combinations. Hence, in the first scheduling, the third path of f_2 is chosen and sent after a delay of one slot. In the second scheduling, since the remaining capacity of slot1 with port 7 has changed, the priority of other paths will be elevated.

It can be seen that although f_4 has the lowest resource occupancy, its lower resource margin implies a higher likelihood of causing port congestion; hence, a lower scheduling priority is given. For flow with multiple possible paths, the algorithm selects the path with the highest resource margin, which can enhance the flexibility of the scheduling. Based on the scheduling results of this example, when the proposed priority sorting mechanism is adopted, flows f_1, f_2, f_3, and f_4 can all be successfully scheduled. In contrast, when using the fixed-path scheduling algorithms such as MSS or ITP, no matter which of the three paths is selected, the third flow will fail to schedule; that is, only the first two flows can be scheduled successfully.

5.2. FPOJS Algorithm

The calculation method of scheduling object priority is given in section A, and the following Algorithm 1 utilizes this method for scheduling. Initially, Algorithm 1 calculates all the shortest paths for each flow based on the network topology, along with their possible offsets, to construct a combination of flow, path, and offset . Subsequently, it computes the priority for each scheduling object. Finally, it performs scheduling in descending order of these priorities, and the scheduling object in each iteration is a combination of flow, path, and offset.

Algorithm 1 Flow–Path–Offset Joint Scheduling (FPOJS) Algorithm

Input: Flow Set F, Topology G
Output: Set of successfully scheduled flows F_suc

```
1:  Initialization SQR[port, t], F_suc, P, Pr[f, path, offset]
2:  for i = 1 to F.num do
3:    for each flow f_k ∈ F do
4:      P = computeAllShortestPaths(f_k, G)
5:      for each p_i ∈ P do
6:        for each offset o ∈ [0, f_k.period − 1] do
7:          if checkSQR(f_k, p_i, o, SQR) == true then
8:            Pr[f_k, p_i, o] =
9:            computePriority(f_k, p_i, o, SQR)
10:         else
11:           Pr[f_k, p_i, o] = −1
12:         end if
13:       end for
14:     end for
15:     [f, p, o] = selectMaxPriority(Pr)
16:     f.offset = o, f.path = p
17:     updateSQR(f, p, o, SQR)
18:     Add f to F_suc
19:     Invalidate priorities for f and update Pr for
        unscheduled flows
20:   end for
21: end for
```

Lines 1–4 calculate the shortest paths of all flows. Lines 5-6 construct the combinations of flow, path, and offset. In line 7, the function $checkSQR(f_k, p_i, o, SQR)$ checks if all ports along the flow path at the given time slot meet the queue resource constraints. If the constraints are satisfied, the scheduling object priority will be calculated; otherwise, the scheduling object is considered as a scheduling failure (lines 7–9). All scheduling objects are then sorted in order of priority from highest to lowest, and the scheduling object with the highest priority is selected as the scheduling target for this round (lines 14–15). Next,

the time slots and ports occupied by this scheduling object are updated, and the scheduling object is considered to be successfully scheduled and deleted from the set of flows waiting for scheduling. The algorithm recalculates other scheduled object priorities and proceeds to the next round of scheduling until there is no flow to schedule (lines 16–18).The original scheduling procedure for this issue initially involved sorting, followed by searching for offsets while treating the routing planning problem as a distinct step. In contrast, the combined routing and scheduling methodology proposed in this paper amalgamates these three steps, utilizing a scoring mechanism based on balanced resource occupancy for selection. This approach achieves a certain reduction in computational complexity, with a time complexity of $O(n^2)$.

5.3. Congestion-Aware Scheduling Algorithm

During scheduling, TSN congestion can be discovered as early as possible by calculating the occupation of resources by flows in real time. Specifically, this involves monitoring the usage of both buffer space and transmission slots in real-time to detect potential bottlenecks. Finding congested nodes and adjusting them properly can improve the scheduling success ratio. Therefore, the congestion-aware scheduling algorithm, Algorithm 2, is proposed. The algorithm is divided into two parts. The first half of the algorithm filters out some ports that may cause congestion, and these ports are often key nodes in the network.This is achieved by utilizing historical data and predictive models to forecast which ports are likely to become congested based on the current flowset and network topology. The algorithm tries to avoid these ports by using other paths of the flows. The second part of the algorithm analyzes the utilization of slot resources of ports after the end of scheduling and selects the ports that are congested in each slot. This involves checking the Shared Queue Region (SQR) of each port for each scheduling slot to determine if the frame length exceeds the maximum transmission unit (MTU). By cross-comparing these ports with those identified as potentially congested, we can determine the transport hub ports that are actually congested.

Algorithm 2 Congestion-Aware Scheduling Optimization

Input: Current flowset F_c, Current topology G_c, Current SQR
Output: Set of rescheduled flows F_r

1: Initialize a list of potential congested ports (PCP)
2: Initialize a list of actual congested ports (ACP)
3: $PCP = ForecastPortUsage(F_c, G_c)$
4: **for** each $f_k \in F_c$ **do**
5: **for** each $pc \in f_k.path$ **do**
6: **if** $pc \in PCP$ **then**
7: Prioritize paths not including ports pc
8: **end if**
9: **end for**
10: **end for**
11: **for** each port $\in G_c$ **do**
12: **for** each slot $\in T_{sched}$ **do**
13: **if** $SQR[port, slot] < MTU$ **then**
14: add $port$ to ACP
15: **end if**
16: **end for**
17: **if** $port \in PCP$ and $port \in ACP$ **then**
18: $SQR[port, :] = 2L_{queue}$
19: **end if**
20: **end for**

In this algorithm, the function $ForecastPortUsage(F_c, G_c)$ in line 3 is used to collect statistics on all ports. Lines 4–9 represent the pre-scheduling phase, which reduces the

priority of flows that may pass through congested ports. Lines 11–16 are used to identify ports where congestion actually occurs. Lines 17–18 are part of the re-scheduling phase, where resources of the congested hub ports are temporarily expanded to twice their original capacity. The expansion method can be either dynamic, where resources are allocated based on real-time congestion levels, or static, where resources are pre-allocated based on predicted congestion patterns.

6. Evaluation

The simulation is conducted within a Python framework, with hardware configurations including an i7-11800H CPU running at 2.3 GHz and 16 GB of RAM. The Python version used in the simulation is 3.9,and the NumPy version is 1.22.0. To validate the performance of the FPOJS algorithm, it is compared against the naive algorithm, the Tabu-ITP algorithm [15], and the MSS algorithm [22]. The naive algorithm sorts flows based on packet size and uses a greedy strategy for offsets. The Tabu list size is set to 20, with a maximum of 300 iterations and 20 repetitions. For the MSS algorithm, the scoring criterion is the ratio of remaining capacity to packet size.

6.1. Simulation Setup

1. Network parameters: In the simulations, the scheduling of CQF is tested using bus, ring, and hybrid network topologies, covering the typical TSN application types. Figure 3 illustrates the bus, ring, and hybrid topology. In the experiments, different TSNs all contains 21 nodes, each with 8 ports. One to three ports of each node connect to the host, and the remaining ports are used for interconnection between nodes. The queue capacity of each port is 8k, the network bandwidth is 1 Gb/s, and the slot length is 125 µs.
2. Flow parameters: The flow period is set between 2 to 9 slots, which is consistent with the properties of flows observed in the real world, and a total of 500 flows are scheduled. Packet sizes are randomly selected within the range of 64 to 1500 bytes. The deadline of flows is set between 2 to 7 milliseconds. The simulation focuses exclusively on unicast flows, while multicast flows can be decomposed into multiple unicast flows.

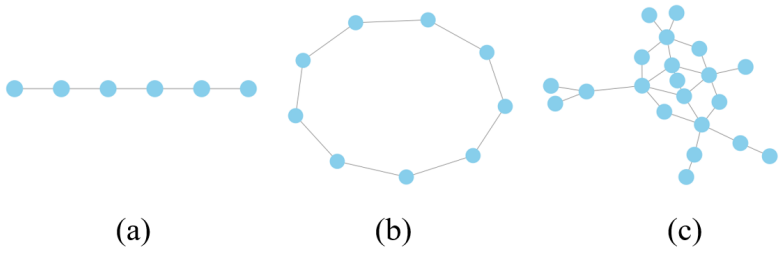


Figure 3. Topology structure: (**a**) bus, (**b**) ring, (**c**) hybrid.

6.2. Simulation Results

6.2.1. Algorithm Performance Under Different Topology Structures

To comprehensively evaluate the performance of our algorithm, it is compared with the naive algorithm, greedy algorithm, Tabu-ITP algorithm, and MSS algorithm across three metrics: scheduling success ratio, execution time, and resource utilization. Figures 4 and 5 show the scheduling success ratios of each algorithm under bus, ring, and hybrid network topologies without considering path constraints, under the conditions of 500 and 800 flows, respectively. The FPOJS algorithm achieves higher scheduling success ratios compared to other algorithms in different topologies and scheduling scales.

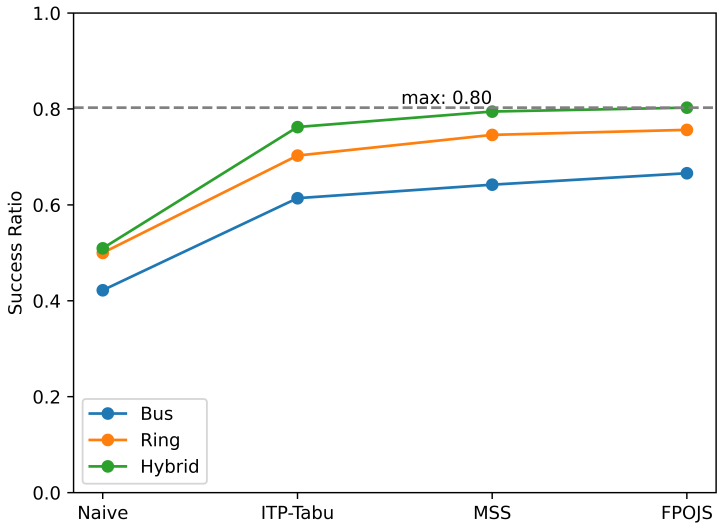


Figure 4. Scheduling success ratio in different topologies with 500 flows.

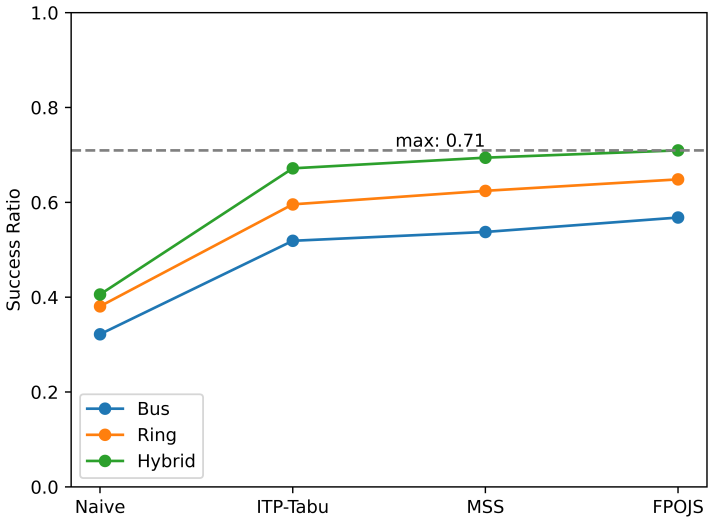


Figure 5. Scheduling success ratio in different topologies with 800 flows.

In the aforementioned experiments, path selection was not incorporated into the scheduling process. The purpose is to demonstrate the success ratio improvement created by the modifications to the priority calculation in the FPOJS algorithm. Across 100 experimental runs, the FPOJS algorithm achieved an average scheduling success ratio that was 1.4% higher than the MSS algorithm and 4% higher than the Tabu-ITP algorithm, with both computation speed and resource utilization surpassing MSS. In subsequent experiments, to demonstrate the validity of incorporating path selection, tests will be conducted in the more complex hybrid topology illustrated in Figure 3. In the hybrid topology, we first test the FPOJS algorithm without using the port resource optimization function. The results are shown in Figure 6. After adding path options, the number of available flows increased by 26%, indicating greater flexibility in scheduling. The scheduling results show that the scheduling success ratio of the FPOJS algorithm is 4% higher than that of MSS without changing the total amount of resources. In large-scale hybrid network scenarios, the FPOJS algorithm has a higher scheduling success ratio. Figure 7 presents a comparison of the scheduling success ratios under varying numbers of nodes. The aforementioned experiments prove that the FPOJS algorithm can be effectively applied to both simple and complex scenarios. Notably, the improvement in scheduling success rate brought about by the FPOJS algorithm is more significant in complex topologies than in simple ones.The overflow of queue resources caused by traffic accumulation is one of the important rea-

sons for the scheduling success rate. Improving the load balancing level can alleviate this phenomenon. Resource utilization can reflect the load balancing level to a certain extent and is positively correlated with the scheduling success rate on the whole. As shown in Figure 8, based on the resource utilization rate of each algorithm with different numbers of nodes in the complex topology, the FPOJS algorithm has better load balancing levels in the complex topology environment. However, due to the inclusion of path selection and an additional step for port resource optimization, it does not exhibit a superior performance in overall computational speed, with an average computation duration of approximately 20% longer than that of the MSS algorithm. Although this has increased the execution time to a certain extent, it is accompanied by significant improvements in scheduling success rate and resource utilization. The comparison of execution time is presented in Figure 9.

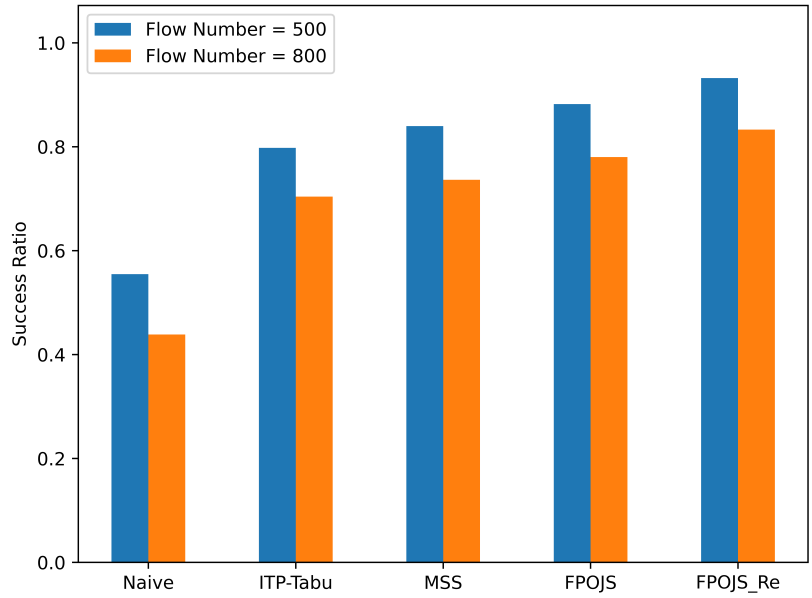


Figure 6. Scheduling success ratios of various algorithms under different loads.

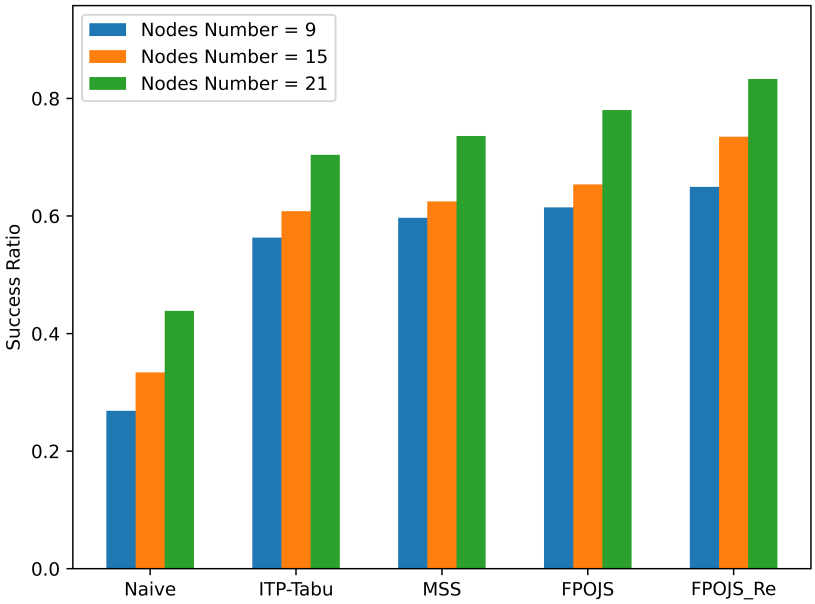


Figure 7. Scheduling success ratios of various algorithms under different node numbers.

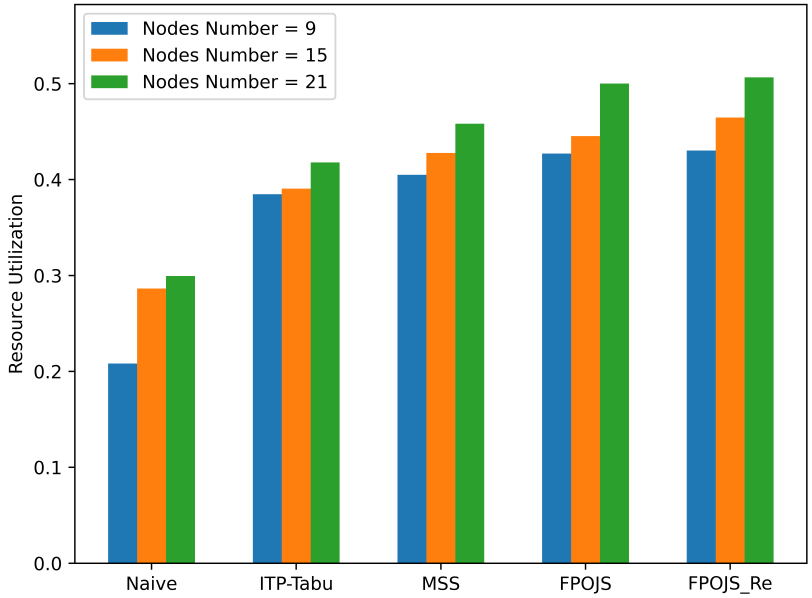


Figure 8. Resource utilization of various algorithms.

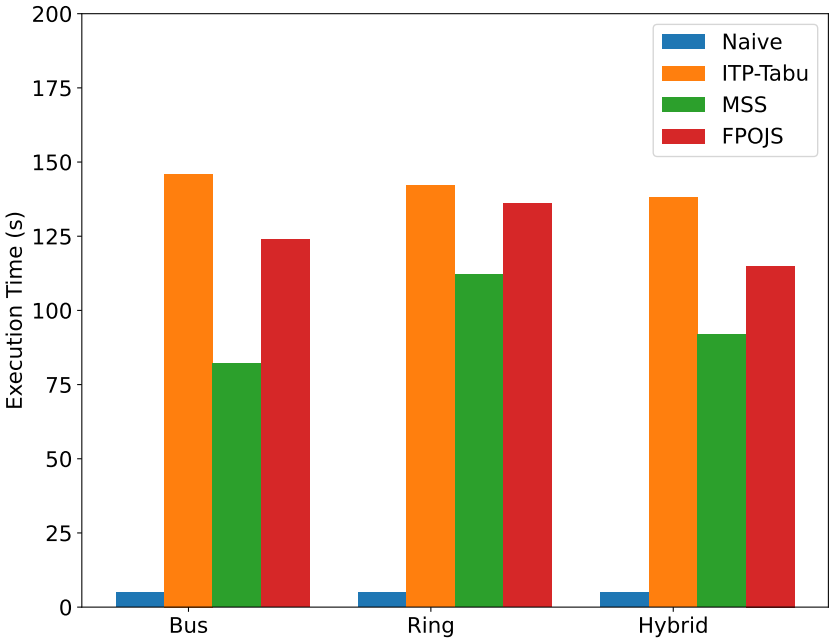


Figure 9. Execution time comparisons of algorithms in different topologies (500 flows).

6.2.2. The Impact of Slot Length and Queue Length on Scheduling

The slot length is related to several factors, including flow period, deadline, link speed, node cache queue length, and network scale. An increase in the node cache queue length can increase the lower bound of the slot length to some extent. Bello et al. [6] and Steiner et al. [7] indicate that an increase in slot length allows nodes to accommodate more packets, thereby increasing the number of flows that can be scheduled. However, the constraint that packets cached from the previous time slot must be entirely transmitted in the next time slot increases the single-hop delay. When single-hop delays accumulate on the path, the total end-to-end delay of the flow increases, resulting in a flow that cannot be scheduled. Additionally, due to the limitations of node hardware resources, the increase in time slot length requires higher memory requirements on the nodes, which in turn increases the complexity and cost of node implementation. More importantly, the precision of gated

control is inversely proportional to the slot length. In a certain traffic period, the range of injection offsets that can be selected by the source system increases with the decrease in the slot length, which not only improves the control accuracy but also increases the search range. The size of the slot length is limited by the bandwidth of the link, and if the time reserved is much longer than the time required to send the queue cache, no data will be transferred to the remaining slots. Figure 10 compares the scheduling success rates among various algorithms for queue lengths of 6000, 8000, 12,000, and 15,000. The comparison reveals that as the queue length increases, more flows can be accommodated in a single scheduling instance, thereby reducing the probability of congestion. This leads to an increase in the number of flows that can be successfully scheduled. Specifically, when the queue length reaches 15,000, the scheduling success ratio of Algorithm FPOJS_RE approaches 100%. This observation underscores the importance of queue length in optimizing scheduling performance and highlights the efficacy of Algorithm FPOJS_RE in managing large-scale flow scheduling with minimal congestion.

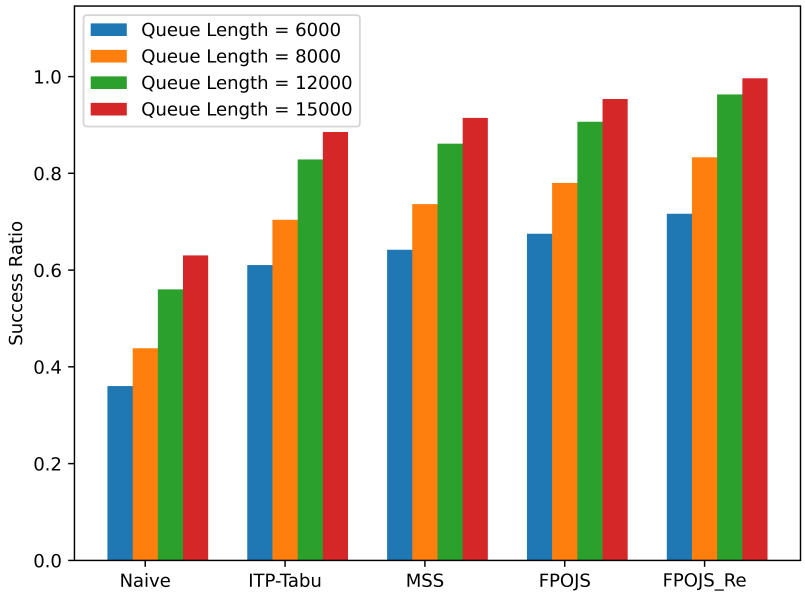


Figure 10. Scheduling success ratios of various algorithms under different queue lengths.

6.2.3. Impact Analysis of Port Resource Expansion on Scheduling Success Ratio

The congestion-aware scheduling algorithm, also known as FPOJS_Re, is an enhanced version of the FPOJS algorithm, which adopts the method of dynamically configuring port resources. In this experiment, the total number of ports is related to the number of access hosts and the topology of TSNs. During scheduling, the algorithm can calculate the congestion degree of the port and sort it. According to the pre-configuration, some ports with high congestion will be expanded by a certain amount. Figure 11 shows the relationship between the number of expanded ports and the scheduling success ratio under the configuration of doubling the capacity of some congested ports. It shows that when the number of expanded ports is 5, the scheduling success ratio increases by approximately 4%; with 10 expanded ports, the scheduling success ratio improves by approximately 8%; and when the number of expanded ports exceeds 20, the increment of the scheduling success ratio becomes more gradual, peaking at around 96%. The experimental results show that selecting the number of expanded ports to be between 5 and 15 can achieve a better balance between scheduling success ratio and resource utilization. Searching for congested ports can lead to an increase in computation time, but in a fixed offline scheduling scenario, the first calculated congested port remains applicable in subsequent scheduling and does not increase the subsequent scheduling time.

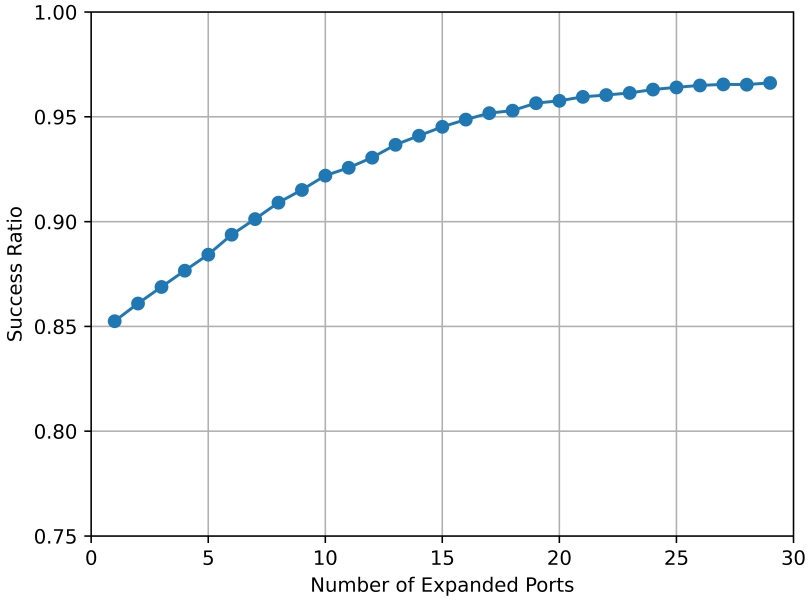


Figure 11. Relationship between the number of expanded ports and the scheduling success ratio.

7. Conclusions and Future Work

In this paper, we presented a novel scheduling object, a combination of flow, path, and offset , and analyzed its importance using an example. We proposed two algorithms to tackle the start time planning issue with the CQF mechanism, namely the FPOJS algorithm and the congestion-aware scheduling algorithm. The FPOJS algorithm performed scheduling in descending order of a well-designed priority, which considered the resource margin and resource requirements of the ports. The congestion-aware scheduling algorithm enhanced the FPOJS algorithm by dynamically configuring port resources. Finally, the experimental results showed that the FPOJS algorithm achieved a 39% improvement in scheduling success ratio over the naive algorithm, a 13% improvement over the Tabu-ITP algorithm, and a 10% improvement over the MSS algorithm. The results also showed that the scheduling success ratio was higher in large-scale heterogeneous network topology. The congestion-aware scheduling algorithm can filter congested resources, and by dynamically expanding only 10% to 20% of the total number of ports in real-time, it can increase the scheduling success ratio by 5% to 10%. It showed great practicability in CQF scheduling applications, which can dynamically allocate port resources.

This study focuses on improving the success ratio in the context of offline scheduling. The online scheduling of CQF mechanisms has higher requirements for computing speed. In future work, we will explore scheduling algorithms that reduce the scheduling computation time.

Author Contributions: Data curation, D.L.; formal analysis, D.L. and Z.Z.; methodology, Z.Z.; writing—original draft, D.L. and Z.Z.; Validation, Z.L.; Formal analysis, Y.W.; Investigation, Z.Z.; Resources, Y.S.; Data curation, J.G.; writing—review and editing, Y.S.; Project administration, Y.S.; Funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are not publicly available, as this is ongoing research.

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

References

1. Deng, L.; Xie, G.; Liu, H.; Han, Y.; Li, R.; Li, K. A Survey of Real-Time Ethernet Modeling and Design Methodologies: From AVB to TSN. *ACM Comput. Surv.* **2023**, *55*, 1–36. [CrossRef]
2. Pei, J.; Hu, Y.; Tian, L.; Li, M.; Li, Z. A Hybrid Traffic Scheduling Strategy for Time-Sensitive Networking. *Electronics* **2022**, *11*, 3762. [CrossRef]
3. Farkas, J.; Bello, L.L.; Gunther, C. Time-Sensitive Networking Standards. *IEEE Commun. Stand. Mag.* **2018**, *2*, 20–21. [CrossRef]
4. Craciunas, S.S.; Oliver, R.S.; Chmelík, M.; Steiner, W. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest France, 19–21 October 2016; ACM: New York, NY, USA , 2016; pp. 183–192.
5. Vlk, M.; Hanzalek, Z.; Brejchova, K.; Tang, S.; Bhattacharjee, S.; Fu, S. Enhancing Schedulability and Throughput of Time-Triggered Traffic in IEEE 802.1Qbv Time-Sensitive Networks. *IEEE Trans. Commun.* **2020**, *68*, 7023–7038. [CrossRef]
6. 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]
7. Steiner, W.; Craciunas, S.S.; Oliver, R.S. Traffic Planning for Time-Sensitive Communication. *IEEE Commun. Stand. Mag.* **2018**, *2*, 42–47. [CrossRef]
8. Quan, W.; Yan, J.; Jiang, X.; Sun, Z. On-line Traffic Scheduling optimization in IEEE 802.1Qch based Time-Sensitive Networks. In Proceedings of the 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Yanuca Island, Cuvu, Fiji, 14–16 December 2020; IEEE: New York, NY, USA, 2020; pp. 369–376.
9. Huang, Y.; Wang, S.; Wu, B.; Huang, T.; Liu, Y. TACQ: Enabling Zero-jitter for Cyclic-Queuing and Forwarding in Time-Sensitive Networks. In Proceedings of the IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021 ; pp. 1–6.
10. Steiner, W. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks. In Proceedings of the 2010 31st IEEE Real-Time Systems Symposium, San Diego, CA, USA, 30 November–3 December 2010; IEEE: New York, NY, USA, 2010; pp. 375–384.
11. 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; ACM: New York, NY, USA, 2016; pp. 203–212.
12. Oliver, R.S.; Craciunas, S.S.; Steiner, W. IEEE 802.1Qbv 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; IEEE: New York, NY, USA, 2018; pp. 13–24.
13. Nayak, N.G.; Dürr, F.; Rothermel, K. Time-sensitive Software-defined Network (TSSDN) for Real-time Applications. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; ACM: New York, NY, USA, 2016; pp. 193–202.
14. 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; IEEE: New York, NY, USA, 2019; pp. 1–6.
15. 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; IEEE: New York, NY, USA, 2020; pp. 616–625.
16. Cheng, Z.; Yang, D.; Zhang, W.; Ren, J.; Wang, H.; Zhang, H. DeepCQF: Making CQF Scheduling More Intelligent and Practicable. In Proceedings of the IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; IEEE: New York, NY, USA, 2022; pp. 1–6.
17. Nie, H.; Li, S.; Liu, Y. Optimized scheduling mechanism based on IEEE 802.1Qch standard in time-sensitive networking. *J. Commun.* **2022**, *43*, 12–26.
18. 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. Inf.* **2022**, *18*, 9312–9323. [CrossRef]
19. 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]
20. Feng, Z.; Gu, Z.; Yu, H.; Deng, Q.; Niu, L. Online Rerouting and Rescheduling of Time-Triggered Flows for Fault Tolerance in Time-Sensitive Networking. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2022**, *41*, 4253–4264. [CrossRef]
21. 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]

22. Guo, M.; Gu, C.; He, S.; Shi, Z.; Chen, J. MSS: Exploiting Mapping Score for CQF Start Time Planning in Time-Sensitive Networking. *IEEE Trans. Ind. Inf.* **2023**, *19*, 2140–2150. [CrossRef]
23. Wang, X.; Yao, H.; Mai, T.; Xiong, Z.; Wang, F.; Liu, Y. Joint Routing and Scheduling with Cyclic Queuing and Forwarding for Time-Sensitive Networks. *IEEE Trans. Veh. Technol.* **2023**, *72*, 3793–3804. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.