applied sciences

MDPI

Article

An Enhanced Routing and Scheduling Mechanism for Time-Triggered Traffic with Large Period Differences in Time-Sensitive Networking

Hongrui Nie [1,*], Shaosheng Li [2,*] and Yong Liu [1]

[1] School of Information and Commnunication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; liuyo@bupt.edu.cn
[2] School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: nie_hy@bupt.edu.cn (H.N.); lss@bupt.edu.cn (S.L.)

Abstract: In the field of the automotive area as well as industrial control, real-time communication requires deterministic delivery with low delay and bounded jitter. Real-time communication in these networks requires transmission schedule and routing, which is an NP-hard problem. In this paper, we present an offline routing and scheduling method based on integer linear programming (ILP), with a flow preprocessing step to explore the period correlation of time-triggered (TT) traffic in time-sensitive networking (TSN). First, a multiperiod flow routing and scheduling algorithm based on flow classification is proposed to improve the scheduling success rate and reduce execution time. The flow classification technique obtained a more fine-grained TT traffic classification, which can be superimposed on any routing and scheduling algorithms. Second, an adaptive period compensation scheduling algorithm based on flow classification is proposed in simple network architecture conditions. The evaluations demonstrate that the proposed algorithms improve scheduling success rate and reduce execution time compared with baseline methods in all test cases. In addition, we can adapt our different proposed algorithms in different network architecture conditions to schedule various flows with different periods and sizes.

Keywords: time-sensitive network (TSN); integer linear programming (ILP); routing and scheduling algorithm; flow classification

Citation: 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. https://doi.org/10.3390/app12094448

Academic Editors: Teen-Hang Meen, Charles Tijus, Po-Lei Lee and Chun-Yen Chang

Received: 6 April 2022
Accepted: 25 April 2022
Published: 28 April 2022

1. Introduction

With the continuous development of network communication, many application scenarios such as industrial IoT, vehicular networks, and avionics networks have increasingly higher requirements for real-time and reliable transmission, while traditional Ethernet can no longer meet the requirements of applications. Consequently, a more optimal design of communication systems is needed to achieve deterministic and low end-to-end transmission delay for a massive amount of time-triggered traffic. Traditionally, industry networks adopted a dedicated field bus [1] to obtain deterministic guarantees for real-time data transmission. What is more, various real-time Ethernet technologies, such as EtherCAT [2] and TTEthernet [3] have been proposed to achieve hard real-time guarantees for Ethernet.

In addition, the IEEE 802.1 TSN task group puts forward the concept of a time-sensitive network (TSN) [4–6], which intends to build a unified interface that can be used in various fields through standardization to support real-time transmission. TSN supports real-time applications with zero packet loss and bounded end-to-end latency. To achieve bounded delay and low jitter transmission, the TSN task group improves a series of standards [7] including IEEE 802.1Qbv supporting time-aware shaper to schedule real-time data in a time-triggered mechanism, whereas TSN standards do not cover specific algorithms for routing and scheduling.

The routing and scheduling problem has been proven to be an NP-hard problem, and there are generally directly solving scheduling and search-based scheduling algorithms to solve such problems. Pahleavan proposes two routing and scheduling algorithms based on heuristic algorithms, i.e., a genetic algorithm [8] and a heuristic list scheduler [9]. The NEH algorithm proposed in [10] is a well-established initialization heuristic in genetic algorithms. Additionally, Arestova discusses a design approach for a hybrid genetic algorithm proposed in [11] which can compete with the well-adapted NEH algorithm. However, all mentioned heuristic algorithms model and optimize routing and scheduling, respectively. Routing selection typically uses the shortest path first (SPF) and load balance first (LBF) algorithms.

Satisfiability Modulo Theories (SMT) and integer linear programming (ILP) mathematical techniques can jointly solve the routing and scheduling problem to meet the timing requirements [12–14]. The most common optimization goals include the shortest path and reducing maximum load balancing to improve scheduling success [15,16]. The path generated by these algorithms can only ensure a low end-to-end latency of newly added traffic while they cannot guarantee the success rate of added traffic and also cannot confirm whether they have a serious impact on scheduled traffic. Yu [17] adds a preprocessing stage based on [13] which removes unnecessary links and generates a subgraph of the network. Additionally, an improved topology pruning is introduced to reduce the scale of the scheduling problem [18]. Inspired by this approach, we introduce a flow preprocessing step before optimizing on the network side. Nevertheless, existing scheduling algorithms for TT traffic do not address the impact of route planning and time scheduling results on scheduling success, the number of schedulable flows, and the future capacity of the network to carry traffic when the differences between different traffic periods are too large [15,19,20].

1.1. Contributions

The motivation of this paper is to resolve the problem of routing and scheduling of TT traffic considering characteristics of flow period satisfying the deterministic transmission requirements for different network scales and network connectivity. In this paper, an enhanced routing and scheduling approach based on flow classification is explored to maintain the quality of service of applications with different periods. In particular, our contributions to this paper are as follows:

- We preprocess the preknown flows and group them based on the period correlation to obtain a delimited set of flows, named flow classification (FC).
- Based on the FC, a multiperiod flows routing and scheduling algorithm (MPFRS-FC) adopts the iterated ILP-based scheduling and routing in order to attain high scheduling scalability by flow group, and incrementally adding a flow group to the scheduled procedure is proposed. Additionally, the link pruning after successful scheduling of each flow group in the MPFRS-FC can effectively reduce the search space and execution time.
- Furthermore, an adaptive period compensation routing and scheduling algorithm based on FC (APCRS-FC) is designed to tackle the simple or loosely coupled network topology carrying complex periodic TT traffic problems, since multiple disjoint paths need to be found in the MPFRS-FC algorithm to minimize the number of unschedulable flows.
- Through evaluations, the paper shows that the proposed algorithm outperforms benchmark routing and scheduling algorithms in terms of scheduling success rate and execution time.

1.2. Outline of the Paper

This remaining paper is structured as follows. The background and related work is discussed in Section 2. The system model is presented in Section 3. In Section 4, we propose MPFRS-FC and APCRS-FC techniques for time-triggered flows to improve scheduling rate and reduce execution time. We present the evaluations of our algorithms with a relevant discussion in Section 5 and the conclusion in Section 6, respectively.

2. Background and Related Work

The IEEE TSN standards released IEEE 802.1 Qbv standard [21] to support the time-triggered mechanism with low bounded latency and jitter guarantees. Before network configuration and operation, a scheduler collects network topo information and timing parameters of time-triggered flows as input and generates a valid scheduling table. Then, it sends down scheduling tables to each switch, and switches periodically perform scheduling table entries to achieve deterministic communication and latency guarantees.

Transmission scheduling in TSN networks to impact real-time performance is a well-researched problem [22–27]. The resource allocation mechanism in TSN can be considered from two aspects in general. The first subproblem is the routing algorithm, i.e., how to select a path from source to destination for each flow. The second subproblem is the scheduling method, i.e., when to send which frame to be sent or forwarded.

Concerning the routing subproblem, the optimal objective is to minimize the number of hops for routing traffic traditionally [28]. However, Laursen et al. proved that the shortest path algorithm is not an excellent routing algorithm for TSN because this approach may result in nonschedulable solutions. Hence, many heuristic routing algorithms are proposed [8–11,29] to determine the route paths with different objective optimization. The work in [30] proposes a K-shortest path heuristic and a metaheuristic randomized adaptive search procedure to solve the joint routing and scheduling problem for time-triggered traffic. Some existing approaches use the Steiner Tree as the default route [24,25]. In the schedule synthesis, a routing and scheduling procedure is devised to improve the schedule success rate. The scheduling procedure is based on the formerly fixed routing, and thus the execution time of the schedule is sharply reduced.

To solve the scheduling problem in TSN, several prior works used mathematical programming Satisfiability Modulo Theories (SMTs) based on solver (Z3) to compute schedules for TTEthernet, profiNET, and IEEE 802.1Qbv [31–34]. Additionally, the results in [12] show that the computational complexity increase exponentially when the number of flows increases linearly. E. Schweissguth et al. [13] observed that prior works separate the flow routing subproblem from the flow scheduling subproblem. The typical solution is to solve the routing subproblem first and then compute the schedule based on fixed routing results. They formulated routing and scheduling jointly as an ILP problem using a Gurobi optimizer to solve the problem. Nayak et al. [15] studied how to improve schedulability and proposed an ILP-based algorithm with a maximum scheduled traffic load (MSTL) metric to determine the routing of each TT flow. Dürr and Nayak [20] address a special problem named no-wait packet scheduling, where the frames are not allowed to wait in the switches. The switch forwards the frame to the next node as soon as it receives the frame. This method avoids the frames jitter by assuming that all flows have the same period, and thus, the problem is restricted to one frame per flow. Jin, Xia et al. [35] proposed a joint algorithm of appropriate message fragmentation and no-wait scheduling to improve the schedule success rate up to 50% further than previous algorithms. Nayak et al. [27] propose an approach that schedules the transmission of time-triggered flows only on the hosts. Moreover, authors of [17,36,37] observed the effect of different routing results on scheduling and recommended joint routing and scheduling algorithms to use proper network resources and to target load balancing.

Atallah et al. [36] considered frame period as a factor of the degree of conflict measured and incrementally added to the schedule; this approach is based on assuming densely connected typologies such that the possible paths between any two nodes outnumber those required for messages exchanged between these nodes. We observed that the above approaches omitted the period correlation between these flows. The approach [36] does not focus on the correlation between traffic periods. We proposed the flow classification (FC) technique which finds the correlation of multiple flows' periods to utilize the same links. Based on FC, we proposed the MPFRS-FC algorithm to generate routes based on traffic period characteristics, which is suitable for relatively complex topologies where there are more than two disjoint paths between two end systems that can be transmitted.

If during route discovery, flows that are not in the same class must share partial path to reach the destination, then the scheduling step can be terminated early and the newly added flow can be placed in the nonschedulable flow set. This process simplifies the complexity of joint routing and scheduling optimization. Then, the APCRS-FC algorithm is not only appropriate for network topologies with simple or loosely coupled structures but also can further improve the schedulability of the nonschedulable flow set output by the MPFRS-FC algorithm.

3. System Model

In this section, we present the network model and the flows specification assumed throughout this paper. We use uppercase letters for sets, lowercase bold letters for vectors, and object-oriented notation for attributes. Table 1 summarizes the notation used.

Table 1. Notations.

	Notation	Description
index	i, j, x, y, z	the index of a node
	k, m	the index of a flow
	$[i, j]$	the index of a link
	l	the index of a link
set	V	the vertex set
	E	the edge set
	ES	the set of all end systems
	SW	the set of all switches
	F	the set of all flows
vector	$\mathbf{src^F}$	source vector
	$\mathbf{dest^F}$	destination vector
	$\mathbf{prd^F}$	period vector
	$\mathbf{td^F}$	transmission duration vector
	$\boldsymbol{\omega}$	transmission window offset vector
decision variable	$\mathbf{r}$	route decision vector
	$\boldsymbol{\varphi}$	time slot offset vector

3.1. Network Model

We denote the time-sensitive network (TSN) as a directed graph $G = (V, E)$, where the set $V = ES \cup SW$ is the set of all nodes (end systems and switches) connected by physical links, and the set $E \subseteq V \times V$ includes a set of full-duplex physical links or edges. Note that each full-duplex link is regarded as two-directional logical links denoted by ordered pairs (v_i, v_j) and (v_j, v_i), where the first element is the source and the latter is the destination. $|V|$ and $|E|$ denote the number of nodes and links, respectively.

A physical link (v_i, v_j) is characterized by the tuple $< (v_i, v_j).r, (v_i, v_j).d >$, where $(v_i, v_j).r$ represents the transmission rate of the link (v_i, v_j), $(v_i, v_j).d$ is the sum of maximum latency which is independent of the scheduling algorithm.

An example of the TSN topology is shown in Figure 1. Each end system executes one or more types of real-time application that periodically sends flows under firm latency constraints. We assume that the whole network is precisely synchronized, that is, as long as the routes and transmission time slots are properly planned, the flows can be transmitted based on the schedule without conflict links in ideal circumstances. The controller has to gather all network information and all flow information and schedule routes and time slots. A feasible route consists of end systems, switches, and links which the flows traverse. The time slot must be reserved for the links within each flow's route.

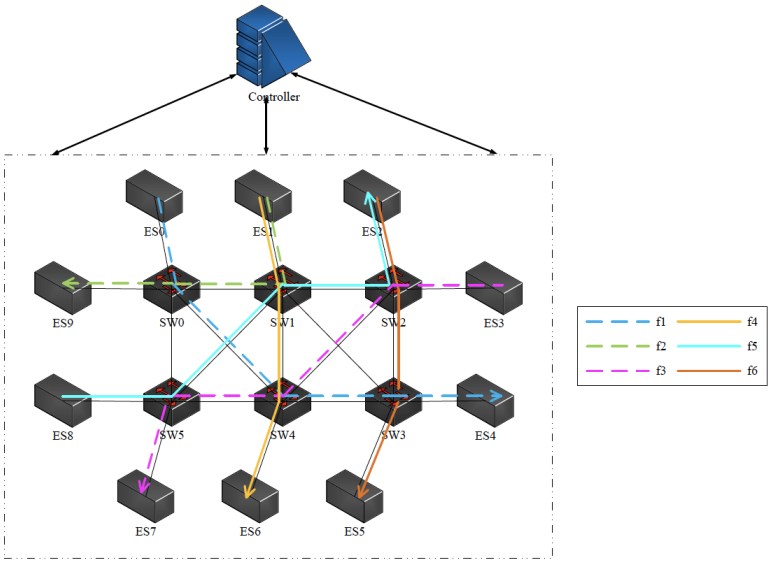


Figure 1. Example TSN topology model $G(16, 21)$ including ten end systems which schedule six types of flows via six switches with multiple no-conflict paths. A controller collects application information and network topology information and reports to scheduler module. The schedule results can be deployed on the TSN devices.

3.2. Flow Specification

The delay-critical applications are modeled as periodic tasks that send recurrent flows grouped in classes. Each flow is transmitted within its period, and the hyper cycle is defined as the least common multiple of the periods of all flows. We consider a set of n *aperiodic* unicast time-triggered flows $F \triangleq \{f_1, f_2, \ldots, f_n\}$. For simplicity, the multicast case can be split into multiple unicast flows with the same source and different destinations. Each flow $f_k \triangleq (f_k.class, f_k.src, f_k.dest, f_k.size, f_k.prd, f_k.ddl, f_k.td) \in F$ is characterized by a 7-tuple, where $f_k.src, f_k.dest \in V$ are the flow's source and destination node, respectively. $f_k.class$ is the flow category which affects the selection of routes. $f_k.size$ defines the frame length in bytes and $f_k.prd$ is the period of the flow. Moreover, $f_k.ddl$ denotes the maximum allowed end-to-end delay. We assume that all flows are transmitted simultaneously over the network. When all flows are released, we can safely restrict our attention to the first cycle, called the hyper cycle. $f_k.td$ denotes the transmission duration of a flow related to the link transmission rate and flow size. Suppose that the size of the flow is 1000 bytes, then the transmission duration of flow sent to 1000 Mb/s link on the port is 8 µs. We assume that the unit for $f_k.prd$ and $f_k.td$ is microsecond.

4. Proposed Solution

We define time-triggered flows with different periods as heterogeneous flows and assume that all links are homogeneous which all have the same characteristics. To facilitate the understanding of our proposed routing and scheduling strategy, we first explain some of the concepts used in it.

Definition 1. *(Scheduling Cycle) The length of schedule denoted by $t^F_{cycle}(p_{i,j})$ is defined as the least common multiple of the periods of all time-triggered flows to be scheduled via the port $p_{i,j}$, which is attached to only one directional link (v_i, v_j).*

$$t^F_{cycle}(p_{i,j}) \triangleq LCM(f_1.prd, f_2.prd, \ldots, f_n.prd) \tag{1}$$

Definition 2. *(Time Slot) The size of one slot denoted by t_{slot}^{F} is defined as the greatest common divisor of the periods of all time-triggered flows to be scheduled.*

$$t_{slot}^{F} \triangleq GCD(f_1.prd, f_2.prd, \ldots, f_n.prd) \tag{2}$$

Definition 3. *(Flow Correlation) The correlation of flow cycles denoted as F_{corr} is defined as the ability of heterogeneous flows to share bandwidth without conflict. We use the number of situations that other flows can be scheduled without conflict as a measure of correlation when one flow already exists in the network. We denote $num(f_m, f_k)$ as the number of situations that accommodated flows f_k can coexist with f_m, assuming that flow f_m has been assigned to be transmitted through one link at zero time slot.*

$$num(f_m, f_k) = \lfloor \frac{f_k.prd - \frac{f_k.prd}{GCD(f_m.prd, f_k.prd)}}{f_k.td} \rfloor \tag{3}$$

Proof of Definition 3. The periods of flow f_m and f_k are defined as $f_m.prd$ and $f_k.prd$, respectively. We assume that the flow f_m has already been scheduled in the network in the earliest iteration. In the overall iterative scheduling algorithm, flow with smaller period is prioritized to reduce the number of iterations, i.e., $f_m.prd \leq f_k.prd$.

We take the sending instant of flow m as the reference zero point to analyze, for flow f_k, how many combinations of time slots can be available to be scheduled on the same link (v_i, v_j) with the guarantee that the two flows do not conflict.

The offsets of the flow f_m and f_k in the first cycle are φ_m and φ_k, respectively, and the transmission cycle constraints $0 \leq \varphi_m < f_m.prd$ as well as $0 \leq \varphi_k < f_k.prd$ are satisfied as shown in (15). Then, the locations of time slots to sending or forwarding flow f_m and f_k are $ts_m = \omega_m \times f_m.prd$ and $ts_k = \omega_k \times f_k.prd + \varphi_k$, where $\mathbf{W_m} = \{\omega_m \in N : 0 \leq \omega_m \leq \frac{t_{cycle}^{\{f_m, f_k\}}}{f_m.prd} - 1\}$ and $\mathbf{W_k} = \{\omega_k \in N : 0 \leq \omega_k \leq \frac{t_{cycle}^{\{f_m, f_k\}}}{f_k.prd} - 1\}$.

The conditions under which collisions occur between two flows transmitting on the same link are

$$\omega_m \times f_m.prd = \omega_k \times f_k.prd + \varphi_k, \tag{4}$$

that is, the contention constraints are not satisfied as shown in (13). The time slot $t_{slot}^{\{f_m.f_k\}}$ is defined as

$$t_{slot}^{\{f_m.f_k\}} = GCD(f_m.prd, f_k.prd). \tag{5}$$

We can obtain that $\exists\ \mathfrak{m}_0, \mathfrak{k}_0 \in \mathbb{N}$, $f_m.prd = \mathfrak{m}_0 \times t_{slot}^{\{f_m.f_k\}}$ and $f_k.prd = \mathfrak{k}_0 \times t_{slot}^{\{f_m.f_k\}}$ are satisfied. Then, the collision condition can be expressed as

$$\varphi_k = (\omega_m \times \mathfrak{m}_0 - \omega_k \times \mathfrak{k}_0) \times t_{slot}^{\{f_m.f_k\}} \tag{6}$$

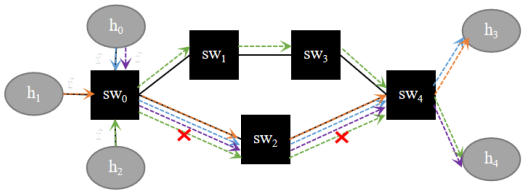
Based on Bezout Theorem, *if a and b are integers, and $gcd(a, b) = d$, then $ax + by$ must be a multiple of d for any integer x, y. In particular, there must exist integers x, y such that $ax + by = d$.* An important corollary is that a sufficient condition for the mutuality of a and b is that there exist integers x and y such that $ax + by = 1$.

Here, $t_{slot}^{\{f_m.f_k\}}$ is the greatest common divisor of $f_m.prd$ and $f_k.prd$, additionally, $\mathfrak{m}_0$ and $\mathfrak{k}_0$ are two mutually prime numbers, i.e., $gcd(\mathfrak{m}_0, \mathfrak{k}_0) = 1$. There exist integers $\tilde{m}$ and $\tilde{k}$ such that $\tilde{m} \times \mathfrak{m}_0 + \tilde{k} \times \mathfrak{k}_0 = 1$. Since the time slot location of the offset must be integer multiples of the time slot, $\forall \chi \in \mathbb{N}, (\omega_m \times \mathfrak{m}_0 - \omega_k \times \mathfrak{k}_0) \times t_{slot}^{\{f_m.f_k\}} = \chi \times (\tilde{m} \times \mathfrak{m}_0 + \tilde{k} \times \mathfrak{k}_0) \times t_{slot}^{\{f_m.f_k\}}$ is satisfied when and only when $\varphi_m = \chi\tilde{m}$ and $\varphi_k = \chi\tilde{k}$.

The offset of flow f_k in the first cycle cannot be chosen from $\{0, t_{slot}^{\{f_m.f_k\}}, 2t_{slot}^{\{f_m.f_k\}} \ldots, f_k.prd - t_{slot}^{\{f_m.f_k\}}\}$. Therefore, the number of available time slot combinations is $num(f_m.f_k) = \lfloor \frac{f_k.prd - \frac{f_k.prd}{t_{slot}^{\{f_m.f_k\}}}}{f_k.td} \rfloor$. The definition is proved. □

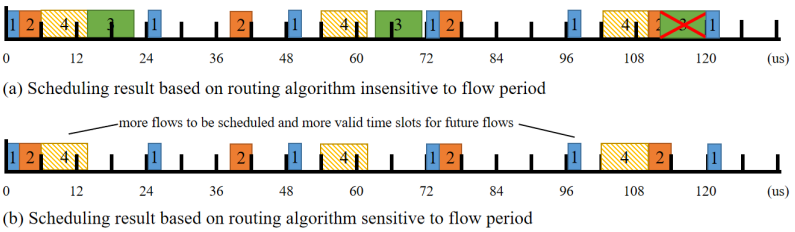
The traditional routing algorithms (e.g., shortest path algorithm, equal-cost multipath algorithm) are focused on finding the shortest route for each flow, which may result in too many flows traversing the same path simultaneously. Here, we fully consider flow characteristics and explore the valid routes that meet the timing constraint $f.ddl$.

To describe how to optimize the routing and scheduling solution, we give an instance of a scenario with four time-triggered flows traversing through switch sw_0 as depicted in Figure 2. The partition of the time slot represents the time resource reserved on the switch's output port, since the flow correlation of flow set $\{f_1, f_2, f_4\}$ is larger than $subset\{f_1, f_2, f_4\} \cup f_3$. From the results of flow correlation, we can find that it is mainly related to the greatest common divisor of flow sets. We can see that f_3 and f_2 congest at time slot 112 in Figure 3b. Additionally, another option for the transmission starting time slot which is sufficient for the transmission of one frame is 26–30 in one cycle, and it will conflict in the following cycle since $GCD(f1, f2, f3, f4) = 1$. There is poor flow correlation between them and it is not suitable to pass through the same route. If the flow is scheduled according to the flow sequence, flow f_3 and other flows cannot be scheduled on the same path without conflict, since they did not have a common divisor and the first frame must be sent within their periods. We consider using the flow classification step to classify the flow first and use a separate path scheduling for flows with periods such as f_3 that seriously affect the scheduling cycle. Without loss of generality, flows with a period that is a multiple of 7 are also added to the same route to schedule. There is no need to recall the routing and scheduling algorithm for global planning.



Flow Information			
Known flow	size	duration	period
f_1	250 bytes	2 us	24 us
f_2	500 bytes	4 us	36 us
f_3	1000 bytes	8 us	49 us
f_4	1000bytes	8 us	48 us

Figure 2. A scenario where four time-triggered flows traverse through one route $[sw_0, sw_2$, and $sw_4]$ generated by SRP routing algorithm.



(a) Scheduling result based on routing algorithm insensitive to flow period

(b) Scheduling result based on routing algorithm sensitive to flow period

Figure 3. Schedule for three flows on the sw_0's output port. The squares symbolize transmission time slots for the respective flow on the link ($sw_0 4$ and sw_2).

The FC step to group the flows by t_{slot} can be considered independent of the network topology to increase the correlation of flows passing through the same link. The whole procedure of FC is presented in Algorithm 1. We introduce three categories to group flows with different periods (lines 5–14). The first group has common divisors other than 1. The flows of this group are further classified based on the additional factors of the third group (lines 15–22). The second group flow cycle has an additional factor $\frac{lcm_{pre}}{lcm_{new}}$. The third group contains the flows that are intermixed with other flow cycles. Flows in this category require a separate path for transmission and cannot share links with other flows. Moreover, flows in the same group can share the same link. The function $generateFlowPartition$ can be easily implemented based on the flow class index. Finally, flows in each group are sorted in ascending order (line 24). The FC is a strategy that can be superimposed on any routing and scheduling algorithms based on finer-grained classification.

Algorithm 1: Flows classification algorithm

Input: flow collection F, flow period set $\mathbf{prd^F}$
Output: An ordered set of flows by flow partitioning $\mathcal{G}$

$lcm_{pre} = LCM(\mathbf{prd^F})$
for $f_i \in F$ **do**
  $lcm_{new} = LCM(\mathbf{prd^F})$
  if $lcm_{new} == lcm_{pre}/f_i.prd$ *or* $f_i.prd$ *is prime* **then**
    $f_i.class = 3$
  end
  if $lcm_{new} == lcm_{pre}$ **then**
    $f_i.class = 1$
    $tmp.append(f_i)$
  else
    $f_i.class = 2$
    $flow\_with\_factor[\frac{lcm_{pre}}{lcm_{new}}] = f_i$
  end
end
for $f_j \in tmp$ **do**
  for $factor \in list(flow\_with\_factor.keys())$ **do**
    if $GCD(factor, f_j.prd)! = 1$ **then**
      $f_j.class = 2$
      $tmp.remove(f_j)$
    end
  end
end
$\mathcal{G} = generateFlowPartition(F)$
$\mathcal{G} \leftarrow sorted(\mathcal{G})$ by ascending order

Then, we focus on the time-sensitive routing and scheduling constraints as well as the optimization objective, which is used for ILP in the next part.

4.1. Routing Constraints

The route allocation is modeled by binary path variable given as follows.

$$r[k,i,j] = \begin{cases} 1, \ if \ f_k \ is \ routed \ over \ link \ (v_i, v_j), \\ 0, \ otherwise. \end{cases} \tag{7}$$

The route for a flow f_i starts at its source $f_i.src$ and ends at its destination(s) $f_i.dest$, hence the number of the outgoing links is exactly one more than the incoming link of $f_i.src$ has to be active, additionally, the number of destinations over the incoming links of the $f_i.dest$ is one.

$if \ i == f_k.src$:

$$\sum_{(i,j)\in E} r[k,i,j] = 1, \ \sum_{(j,i)\in E} r[k,j,i] = 0 \tag{8}$$

$if \ i == f_k.dest$:

$$\sum_{(j,i)\in E} r[k,j,i] = 1, \ \sum_{(i,j)\in E} r[k,i,j] = 0 \tag{9}$$

For the intermediate nodes v_s eligible for forwarding f_k, the flow conservation constraint needs to be satisfied because of the unicast model.

$$\forall v_s \in V \setminus (f_k.src \cup f_k.dest): \sum_{l_1 \in \underrightarrow{v_s}} r[k, l_1] - \sum_{l_2 \in \underleftarrow{v_s}} r[k, l_2] = 0. \tag{10}$$

Here, $\underrightarrow{v_s}$ and $\underleftarrow{v_s}$ are the ingoing link and outgoing link of node v_s eligible for routing f_k.

To prevent routing loops in the same flow, the flow f_k can be forwarded by any node at most once, i.e., each switch receives flows from at most one port.

$$\forall f_k \in F: \sum_{(v_i, v_j) \in E} r[k, i, j] \leq 1 \tag{11}$$

4.2. Schedule Constraints

We introduce integer decision variable $\varphi[k, i, j] \in [0, f_k.prd]$, which is the offset of the time slot occupied by f_k in the first cycle with respect to the zero time slot on link (v_i, v_j). Another integer variable $\omega[k, i, j] \in [0, \frac{t^F_{cycle}(p_{i,j})}{f_i.prd} - 1]$ is the offset of the transmission window, which is the positive integer multiple of the flow period $f_k.prd$. Hence, we can locate time slot position at the beginning of the transmission of the flow f_i on link (v_i, v_j) as $ts[k, i, j] = \varphi[k, i, j] + \omega[k, i, j] \times f_k.prd$. We denote $TS \triangleq \{ts[k, i, j] | f_k \in F, (v_i, v_j) \in R_k\}$. To ensure that the constraints on scheduling decision variable work when link (v_i, v_j) is occupied by flow f_k, the basic constraint is

$$\varphi[k, i, j] \leq M_1 \times r[k, i, j], \forall f_i \in F, \forall (v_a, v_b) \in E \tag{12}$$

where M_1 is an arbitrary but sufficiently large constant.

Contention Constraints: To achieve zero congestion transmission, it is required that all assigned time slots are nonoverlapping. For any two flows f_m, f_k, no flow can be sent to the same link at the same time, otherwise, a conflict occurs. The forward time of outgoing ports of each switch does not conflict. We have the condition

$$\begin{aligned} &\forall f_m, f_k \in F(m \neq k),\ \forall (v_i, v_j) \in E, r[m, i, j] + r[k, i, j] \geq 2: \\ &\varphi[m, i, j] + \omega[m, i, j] \times f_m.prd \geq \varphi[k, i, j] + \omega[k, i, j] \times f_k.prd + f_k.td, \\ &\textbf{or} \quad \varphi[k, i, j] + \omega[k, i, j] \times f_k.prd \geq \varphi[m, i, j] + \omega[m, i, j] \times f_m.prd + f_m.td. \end{aligned} \tag{13}$$

Transmission Cycle Constraints: The flow must be transmitted during its cycle and cannot be confused with the next cycle. The transmission start time of the flow f_k on the link v_i, v_j is constrained by:

$$\forall f_k \in F, \forall (v_i, v_j) \in R_k: 0 \leq \varphi[k, i, j] \leq f_k.prd - f_k.td, \tag{14}$$

$$\forall f_k \in F, \forall (v_i, v_j) \in R_k: \varphi[k, i, j] + \omega[k, i, j] \leq M \times r[k, i, j] \tag{15}$$

Equation (16) ensures that $\varphi[k, i, j]$ and $\omega[k, i, j]$ are invalid when the link (v_i, v_j) is not reserved for flow f_k.

Path Order Constraints: Only when the flow is received by the current node, the previous node can start to transmit the next flow. Hence, the constraint is

$$\begin{aligned} &\forall f_k \in F, \forall (v_x, v_y), (v_y, v_z) \in R_k : \\ &\quad \varphi_{k,y,z} \geq \varphi_{k,x,y} + f_k.D - M \times (1 - r[k,y,z]) \end{aligned} \tag{16}$$

where $f_k.D$ is the one-hop delay of flow f_k, i.e., $f_k.D = f_k.td + d_{prop} + d_{proc}$. This inequality is meaningful only when the flow f_k passes through the link (v_y, v_z), i.e., $r[k,y,z] = 1$.

No-wait Forwarding Constraints: We assume that each switch forwards a data frame immediately after it is received, while satisfying the path constraint above.

$$\begin{aligned} &\forall f_k \in F, \forall (v_x, v_y), (v_y, v_z) \in R_i : \\ &\varphi_{k,y,z} + \omega_{k,y,z} \times f_k.prd = \varphi_{k,x,y} + \omega_{k,x,y} \times f_k.prd + f_k.td \end{aligned} \tag{17}$$

4.3. Application Constraints

The flow f_k must be fully received by the destination $f_k.dest$ within the deadline $f_k.d$. The maximum end-to-end latency constraint is defined as follows:

$$\begin{aligned} &\forall (v_a, v_b) \in R_k, v_a = f_k.src, \forall (v_x, v_y) \in R_k, v_y = f_k.dest : \\ &f_k.l \triangleq (\varphi[k,x,y] + \omega[k,x,y] \times f_k.prd) - (\varphi[k,i,j] + \omega[k,i,j] \times f_k.prd) \\ &f_k.l \leq f_k.ddl - f_k.td \end{aligned} \tag{18}$$

Here, we define $f_k.l$ as the communication latency of a flow f_k from the source to destinations.

4.4. Objective Function

Flow scheduling needs to optimize transmission performance while meeting the above constraints. Therefore, we construct a cost function to evaluate the routing and scheduling scheme, which consists of different optimization objective functions. For each flow f_k, its best route $best\_route(f_k)$ is defined by

$$\begin{aligned} best\_route(f_k) = \min\{ &\sum_{f_m \in \{F-S\}} \sum_{f_k \in S} \frac{1}{F_{corr}(f_m, f_k)} \\ &+ K \times \sum_{(i,j) \in E} r[k,i,j] \} \end{aligned} \tag{19}$$

which considers each flow have a greater number of scheduling situations and also introduce a user-defined positive integer to limit the hops of each route. The FC-based flow correlation objective intends to carry more flows in the network, additionally, flow classification and sorting can reduce the execution time of the routing and scheduling method.

The objective function of the scheduling model is to minimize the end-to-end flow latencies and the sending start time of the flow, which is defined as

$$schedule(\boldsymbol{\varphi}) = \sum_{f_k \in F} f_k.l + \varphi[k, R_k[0]] \tag{20}$$

We optimize the sending time of each flow to make the remaining time slots more available. It breaks up a large number of time slices so that some flows with large frame size cannot be scheduled if the model does not consider that.

The overall procedure of the multiperiod flow routing and scheduling algorithms with flow classification (MPFRS-FC) is shown in Algorithm 2. Given a TSN network G and a flow set F, the output is a mapping $\mathbf{S}[f_k, R[k]]$ of routes and transmission time slots. In the initialization step, MPFRS-FC groups call the flow classification function so that the

paths can be better shared for routing without conflict (line 1). The schedule is built over $num_F = len(\mathcal{G})$ iterations and each iteration handles a flow group. The ILP constraints are generated for $\mathcal{G}_i$, where $i = 0, 1, \ldots, num_F - 1$. The function ILP searches for a feasible solution that satisfies the time-sensitive routing, scheduling, and application constraints. Since the next group of flows cannot use the link contained in the path of the current group, we remove the links which the current group of flow passes through from the graph G and then find the route for the next group of flows (line 11) if the routing or schedule results are empty, i.e. it is infeasible for that flow to be scheduled in the current network.

Algorithm 2: Multiperiod flow routing and scheduling algorithms with flow classification

Input: network architecture G, flow collection F
Output: Mapping of paths to transmission time slots $S_{(path,t)}$, unscheduled flow set **U**

1 $\mathcal{G} \leftarrow flows\_preprocessing(F)$
2 Init **S**, **U**
3 **for** $j \in \{1, 2, 3\}$ **do**
4 Generate constraints for this group of flows $G[j]$
5 $\Xi_1 \leftarrow$ routing constraints (7)–(11)
6 $R[k] = ILP(\Xi_1)$
7 **if** $R[k] \neq \emptyset$ **then**
8 $\Xi_2 \leftarrow$ time-sensitive constraints (12)–(18)
9 $S_f^* = ILP(\Xi_2)$
10 **if** $S_f^* \neq \emptyset$ **then**
11 $\mathbf{S}[k, R[k]].append(S_f^*)$
12 $G \leftarrow G - R[k].edge$
13 **end**
14 $\mathbf{U}.append(f_k)$
15 **else**
16 $\mathbf{U}.append(f_k)$
17 **end**
18 **end**

Furthermore, we found that in some topologies or some network localities, we could not find multiple disjoint paths and could not implement the MPFRS-FC algorithm to improve the schedule success rate. We can use the MPFRS-FC algorithm to quickly locate groups of unschedulable flows, but we cannot schedule them without conflicting routes.

Therefore, an adaptive period compensation routing and scheduling algorithm with flow classification (APCRS-FC) is designed to handle the simple network architecture. There may not be more than one route to pick between the two end systems in this condition. We add optimal compensated value Δ based on the original period. Flows with class 3 can be carried on the same path as flows with class 1 or 2 in that way. This strategy increases a certain delay but can meet the constraints of its deadline.

As shown in Algorithm 3, the flow preprocessing step is first completed with the setting of the flow set (line 1). We introduce an adaptive variable to compensate for the period that can be scheduled in the current network and is within the deadline for that traffic. The following process is similar to the MPFRS-FC. If the compensated flows cannot be scheduled in this way, we remove them from the flow set to ensure the normal scheduling of other traffic, without wasting search time and space on that unnecessary scheduling.

Algorithm 3: An adaptive period compensation routing and scheduling with flow classification

Input: Flow collection F, period vector $\mathbf{prd^F}$
Output: Mapping of paths to transmission time slots $S_{(path,t)}$, unscheduled flow set $\mathbf{U}$

$\mathcal{G} \leftarrow flows\_preprocessing(F)$
Init $\mathbf{S}, \mathbf{U}$
for $j \in \{1, 2\}$ **do**
  for *flow in* $\mathcal{G}[3]$ **do**
    $flow.prd = flow.prd + \Delta$
    $\Xi_0 \leftarrow flow.prd + \Delta$ is a non-one factor of $\mathbf{prd}^{\mathbf{F}\backslash\{\mathbf{flow}\}}$
    $\Xi_0 \leftarrow \Delta \in Z^+$
    $update(flow.class)$
  end
  $\Xi_1 \leftarrow$ routing constraints (7)–(11)
  $\Xi_2 \leftarrow$ time-sensitive constraints (12)–(18)
  $S_{(path,t)}, \mathbf{U} = ILP(\Xi_0, \Xi_1, \Xi_2)$
  if $S_f^* \neq \emptyset$ **then**
    Flows in $\mathcal{G}[3]$ are scheduled;
    return $S_{(path,t)}, \mathbf{U} = \emptyset$
  else
    FLows in $\mathcal{G}[3]$ cannot be scheduled;
    $F = F - \mathcal{G}[3]$
    $R[k] = ILP(\Xi_1)$
    if $R[k] \neq \emptyset$ **then**
      $\Xi_2 \leftarrow$ time-sensitive constraints (12)–(18)
      $S_f^* = ILP(\Xi_2)$
      if $S_f^* \neq \emptyset$ **then**
        $\mathbf{S}[k, R[k]].append(S_f^*)$
      end
      $\mathbf{U}.append(f_k)$
    else
      $\mathbf{U}.append(f_k)$
    end
  end
end

5. Evaluations

5.1. Experiment Setup

We implemented our proposed mechanism in a Python-based framework. We used networkx for the modeling of network topologies and generating different traffic patterns. All of the ILP instances were solved in Gurobi Optimizer (version 9.5.1, 64-bit Windows), which offers a Python API for the integration. The hardware platform is a PC with an Intel Core i7-8750H CPU running at 2.20 GHz and 8GB RAM. The network topologies in test cases consist of four synthetic network topologies shown in Figure 4.

We define three flow groups depicted in Tables 2–4. Each flow is generated by a random and independent selection of groups with their probabilities. The source and destination of each flow were randomly selected from end systems to cover the unicast and multicast broadcast.

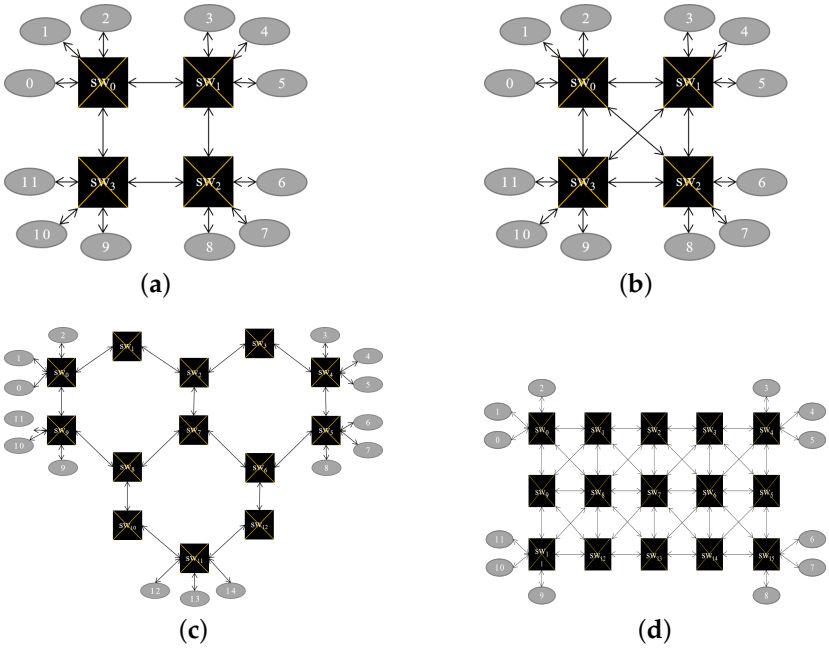


Figure 4. Synthetic topology test cases. (**a**) A small ring topology. (**b**) A small mesh topology. (**c**) A medium ring topology. (**d**) A medium mesh topology.

Table 2. Test group 1 of candidate flows.

Flow Index	F.prd/μs	F.td/μs	Probability
F1	100	1	1/5
F2	200	2	1/5
F3	40	4	1/5
F4	80	8	1/5
F5	160	8	1/5

Table 3. Test group 2 of candidate flows.

Flow Index	F.prd/μs	F.td/μs	Probability
F1	100	1	1/4
F2	200	2	1/4
F3	400	2	7/40
F4	49	1	1/40
F5	50	1	3/10

Table 4. Test group 3 of candidate flows.

Flow Index	F.prd/μs	F.td/μs	Probability
F1	143	1	1/40
F2	130	1	1/40
F3	500	2	1/4
F4	100	2	9/20
F5	250	1	1/4

We evaluated the performance of the MPFRS-FC algorithm, APCRS-FC algorithm, and baseline algorithms. SPRF-EtoED is the shortest path first routing and time slots scheduling by minimizing end-to-end delay. The TSN standard on path control and reservation recommends the constrained shortest path first routing scheme for the transmission of time-sensitive flows in 2016 [38]. LBF-EtoED is the load balance-based first routing and

time slots scheduling by minimizing end-to-end delay [39,40]. All the above are solved by the ILP model. These techniques are adapted to support no-waiting scheduling. Under the conditions of the same number of flows, network topology, and the same flow group, we generated 100 patterns on the three routing and scheduling algorithms and conducted statistics on the scheduling results separately. The comparison is conducted in terms of runtime and the success scheduling rate. SR, SM, and MR model instances were run in Gurobi Optimizer with a time limit of 1 min, MM model instance was set at a time limit of 3 min, and the solution number limit of each model was set to one. Each model finishes with one of the following three possible results:

- Optimal: One suboptimal schedule was found (scheduling as much flow as possible is the pursuit of this paper rather than the optimal solution).
- Timeout: The time limit was reached before the schedule was found.
- Infeasible: No schedule was found with the given flows.

As shown in Figure 5, we simulate four algorithms followed by this workflow. The input is network topology and flow requirements selected from four networks and three flow groups. The flow set generation module generates K random flows based on our given generation rules. The topology and flow resolution module generates the related network graph matrix, vertex association matrix, edge association matrix, and several other data structures that aid in the computation. The routing and scheduling algorithm selection module chooses a specific simulated algorithm to generate route and schedule constraints in the constraint generation module. The constraint addition and solver module calls the Gurobi optimizer to obtain routing and scheduling results. Each simulation scenario is iterated 100 times to measure the performance of the scheduling results.

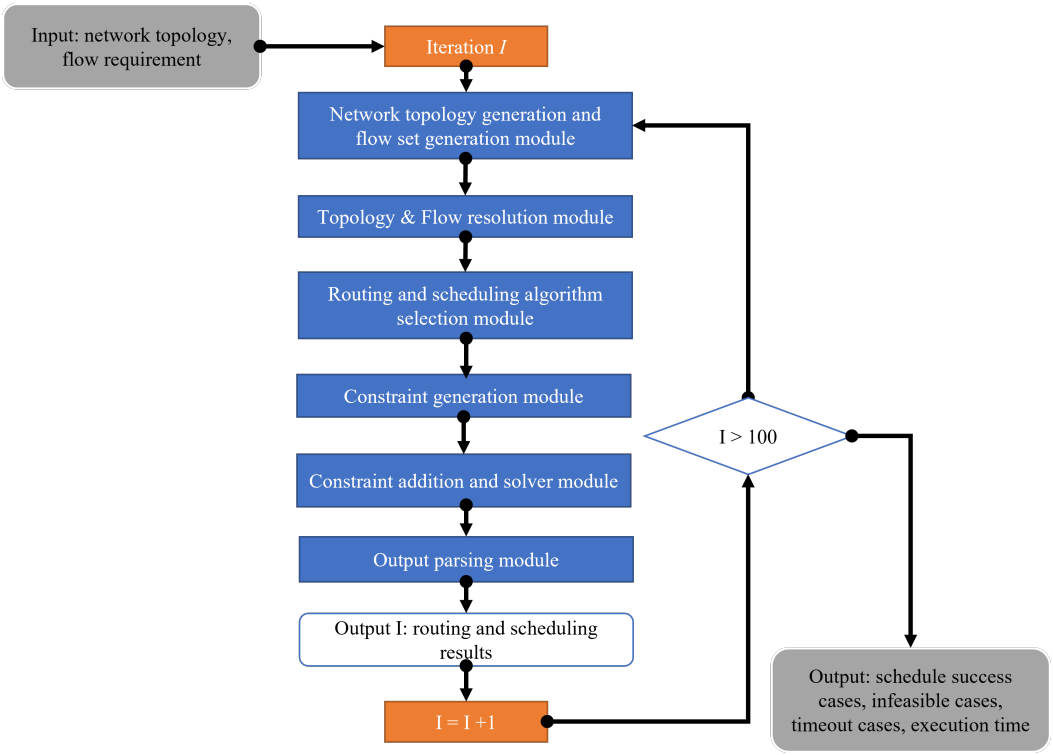


Figure 5. The overall workflow of simulation.

5.2. Performance Evaluation

The performance of the MPFRS-FC and APCRS-FC algorithms are evaluated in comparison with SPRF-EtoED, LBF-EtoED routing, and scheduling techniques. These techniques are adapted to support no-waiting scheduling. The comparison is conducted in terms of the success rate, execution time, and the sum of end-to-end delays. In the following, 12 experiments consisting of 100 test cases are presented.

All success rate results are shown in Figure 6. The orange part of the bars indicates optimized solution, the blue part indicates timeout terminations, and the green part indicates infeasible. In general, when the flow scale is small and the network scale is small, the four technologies can achieve a high scheduling success rate. As the flow scale increases, the flow correlation decreases, the percentage of blue (timeout) cases for the LBF-EtoED algorithm gradually increases. The MPFRS-FC algorithm has the least number of timeouts, i.e., it always gives a clear scheduling result within the range of time limit. In particular, it demonstrates better scheduling performance than the APCRS-FC, SPRF-EtoED, and LBF-EtoED algorithms in highly connected network topologies shown in Figure 6d. Nevertheless, in a relatively simple network topology scenario, the scheduling performance of the APCRS-FC algorithm is optimal. For a more detailed discussion on the algorithm performances, we cover the following points in detail: the impact of traffic load, traffic type, network load, and network type on the performance, respectively.

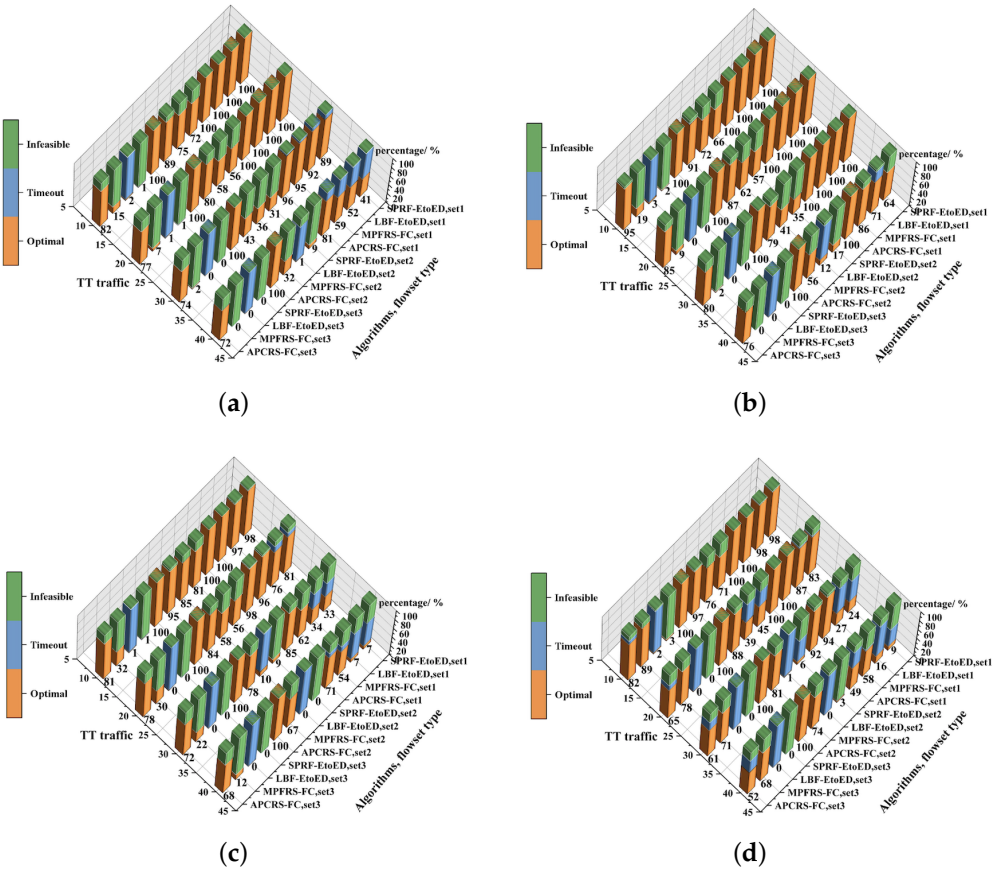


Figure 6. Percentage distribution of optimal, timeout, and infeasible instances of SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for different network topos and different number of flows. (**a**) Topo 1: small ring. (**b**) Topo 2: small mesh. (**c**) Topo 3: medium ring. (**d**) Topo 4: medium mesh.

5.2.1. Performance versus Traffic Load

We compare the execution time of the four techniques for different traffic loads by varying the number of flows while fixing the network type and network size. The number of flows ranges from 10 to 40 flows sent by 12 end systems exchanging the flows over four bridges, and the network type is SM. The maximum execution time and average execution time for 100 test cases are shown in Figure 7. The spherical marker indicates the maximum execution time and the square marker indicates the average execution time. The y-axis shows the execution time of successfully optimal cases, hence the terminated cases due to infeasibility and timeout do not contribute to those computed. Almost low execution times can be achieved by MPFRS-FC and APCRS-FC algorithms at the millisecond level, while

the execution time of the SPRF-EtoED and LBF-EtoED algorithms increases significantly and steeply with the number of flows, even at small network sizes. Moreover, it can be noticed that the speed-up gain increases with a larger number of flows. Furthermore, we noticed that there is also a difference in the performance with the same amount of flows and different types of flow types. In flow group 1, $num(F_{g_1})$ is larger than flow group 2, since we set a special flow type in flow group 2. This not only affects the success rate of scheduling, but we can also see from the Figure 7b that it has a significant impact on the execution time of the SPRF-EtoED and LBF-EtoED algorithms.

In test cases on a slightly larger network scale, we find that the performance gain at runtime is more pronounced. The execution time results in the MM network are shown in Figure 8. The number of flows ranges from 10 to 40 flows sent by 12 end systems exchanging the flows over 15 bridges, and the network type is MM. We set the timeout limit in this test case as 180 s (the yellow band in Figure 8b). For example, the LBF-EtoED technique takes over 180s on average to schedule 40 flows, and it cannot tackle problems of such size. Almost low execution times can also be achieved by MPFRS-FC and APCRS-FC algorithms at the millisecond level in the MM network.

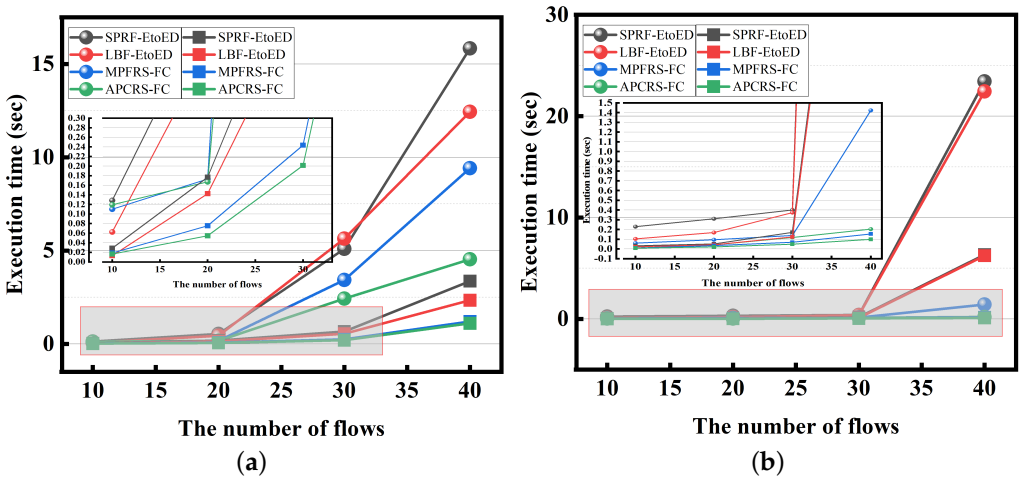


Figure 7. Maximum execution time and average execution time for the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 synthetic test cases with different number of flows, tested in SM network. (**a**) Flows chosen from flow group 1. (**b**) Flows chosen from flow group 2.

Comparisons of the success rate and the end-to-end delay for four flow setups, 10, 20, 30, and 40 flows in the SM and MM networks, are shown in Figures 9 and 10, respectively. SPRF-EtoED and LBF-EtoED obtained better the sum of end-to-end latency metrics, nevertheless, it can be checked in Figure 6 that the majority of unsolved cases by LBF-EtoED were terminated due to the timeout constraint, and those by SPRF-EtoED were terminated due to the infeasible constraint. When the network scale is not large, all algorithms can converge, while when the network scale and flow scale increase, SPRF-EtoED and LBF-EtoED techniques cannot solve the majority of test cases shown in Figure 10a,b. As with the SPRF-EtoED technique, too many flows try to traverse the same path at the same time, which arouses massive infeasible instances. Additionally, the slow exploration of extensive search space by the LBF-EtoED technique can be optimized by the exploration of a limited search space. In addition, LBF-EtoED effectively improves the scheduling performance, while it still cannot further improve the schedulability when the traffic load increases. LBF-EtoED achieves indistinguishable scheduling for flows with different sizes and different cycles, while it does not consider the characteristics of the flow and the combined flows scheduling on the same links. For the highly utilized test cases (40 flows), both proposed algorithms were able to solve 80% of the test cases. APCRS-FC could solve the nearest of

all the test cases in four flow setups, while it increases the end-to-end delay of partial flows within its deadline.

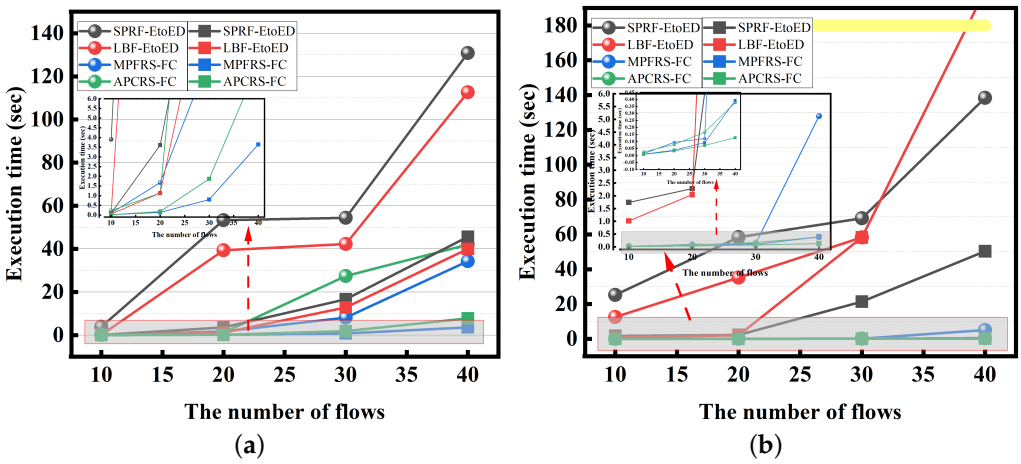


Figure 8. Maximum execution time and average execution time for the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 synthetic test cases with different number of flows, tested in MM network. (**a**) Flows chosen from flow group 1. (**b**) Flows chosen from flow group 2.

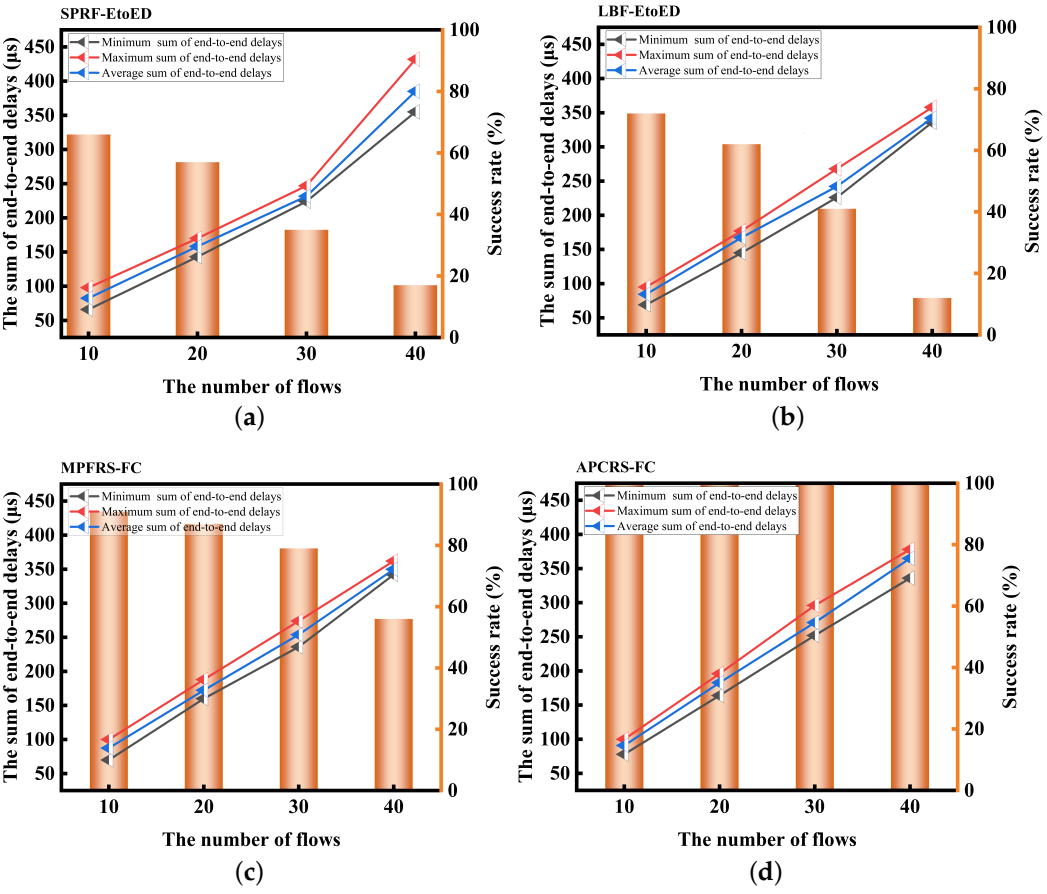


Figure 9. Success rate and the sum of end-to-end delays of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 synthetic test cases with different number of flows chosen from flow group 2, tested in SM network. (**a**) Technique 1: SPRF-EtoED. (**b**) Technique 2: LBF-EtoED. (**c**) Technique 3: MPFRS-FC. (**d**) Technique 4: APCRS-FC.

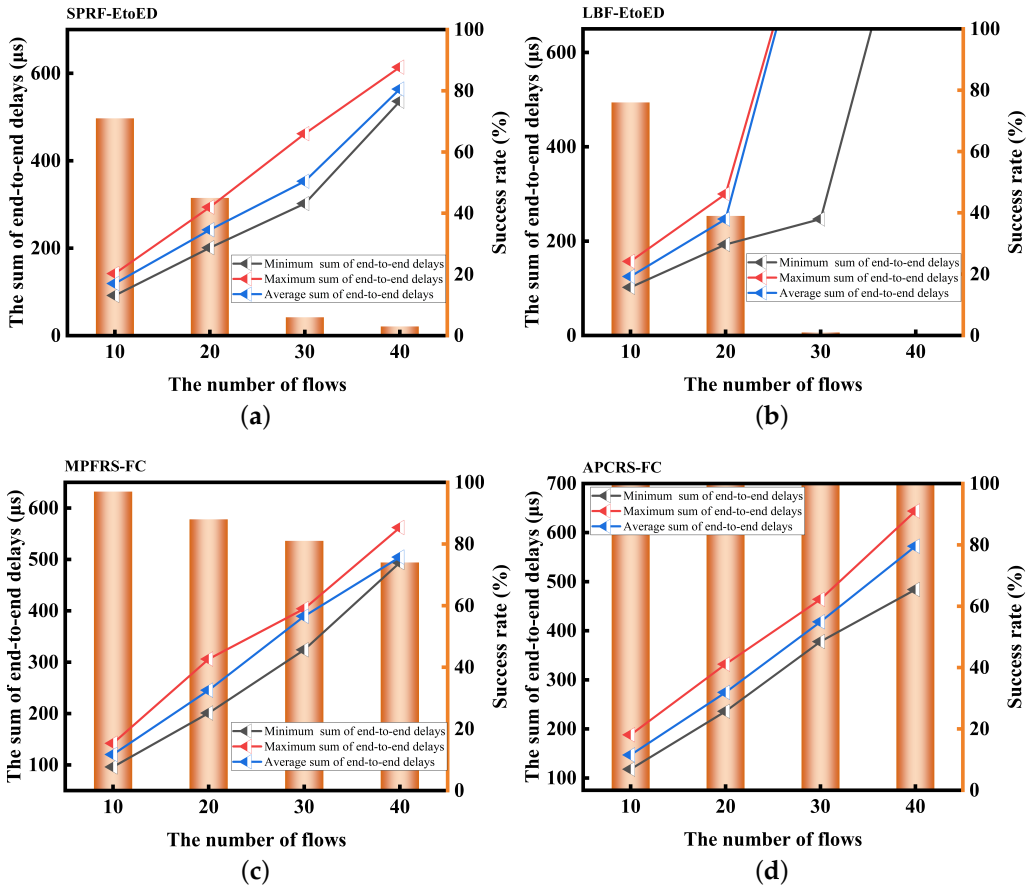


Figure 10. Success rate and the sum of end-to-end delays of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 synthetic test cases with different number of flows chosen from flow group 2, tested in MM network. (**a**) Technique 1: SPRF-EtoED. (**b**) Technique 2: LBF-EtoED. (**c**) Technique 3: MPFRS-FC. (**d**) Technique 4: APCRS-FC.

5.2.2. Performance versus Traffic Type

In this experiment, we compare the scheduled results, execution time, and the sum of end-to-end delays of the four techniques for different traffic types. In total, 100 test cases are handled with 20 flows in the MM network. The scheduled results and execution time results are shown in Figure 11, whereas a comparison of the sum of end-to-end delays is shown in Figure 12. The *x*-axis represents the flow group index and its below corresponds to the algorithm. The average periods and DDL constraints of flows in flow group 1 are larger than flows in flow group 2, while smaller than flows in flow group 3. The flow period correlation decreases sequentially. There is not much variability in the periods in flow group 1 and the existing algorithms can already solve most of the problems. However, for applications in flow group 2 or 3, these exiting techniques are no longer able to tackle them. Contrarily, the results demonstrate the well scalability of the MPFRS-FC and APCRS-FC. Figure 11 shows the superior success rate and faster execution time of the proposed technique over SPRF-EtoED and LBF-EtoED. We noticed that the M algorithm can solve the vast majority of scheduling problems and gives the result of whether or not scheduling is possible in a relatively short time in scenarios where network connectivity is high and flows are poorly correlated. Additionally, it does not have a significant performance loss in terms of end-to-end delay compared with APCRS-FC. The APCRS-FC algorithm takes time the exploration of the compensation factor, resulting in a large number of timeouts when the flow gap is relatively large. Since the terminated cases due to the time limit and infeasible cases do not contribute to the sum of end-to-end delays, SPRF-EtoED has superior overall end-to-end delay performance.

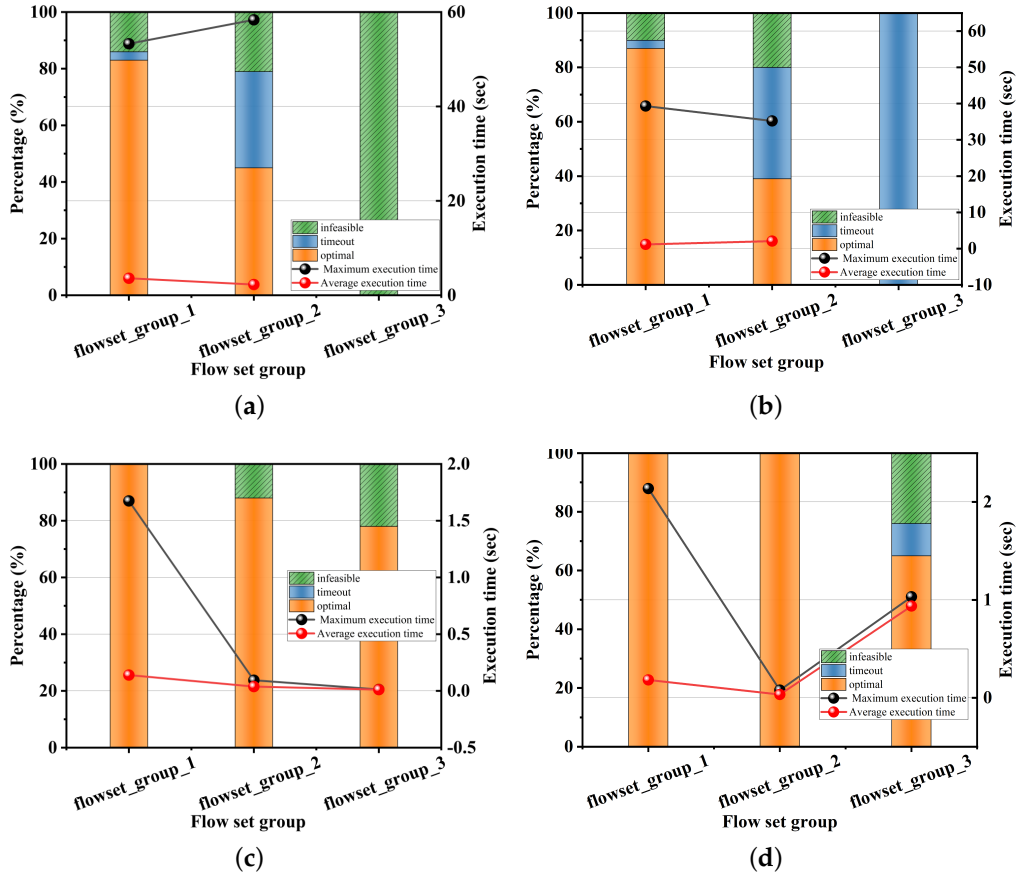


Figure 11. Percentage of solved, timeout, and infeasible schedules and execution time of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 test cases with 20 flows, tested in MM network. (**a**) Technique 1: SPRF-EtoED. (**b**) Technique 2: LBF-EtoED. (**c**) Technique 3: MPFRS-FC. (**d**) Technique 4: APCRS-FC.

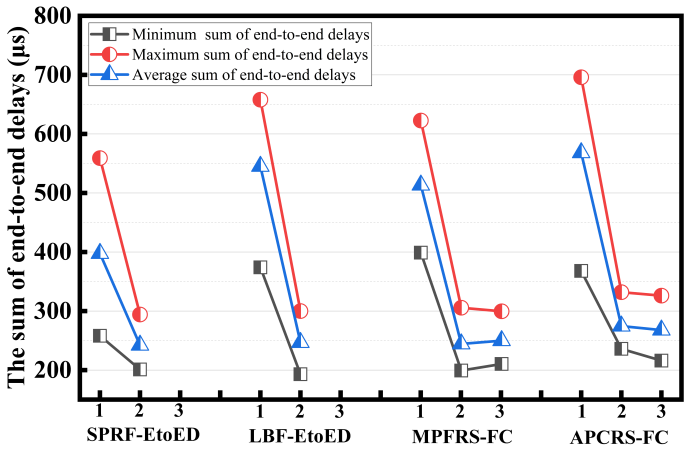


Figure 12. The sum of end-to-end delays of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 test cases with 20 flows, tested in MM network.

5.2.3. Performance versus Network Scale

In this experiment, we compare the schedule results and the execution time of the four techniques for different network scales. In total, 100 test cases are handled with a number of bridges and links (4, 32), (4, 36), (13, 60), and (15, 100). The results are shown in Figure 13. It can be noticed that SPRF-EtoED and LBF-EtoED not only showed a massive number of timeouts but also showed a rapid increase with larger networks, whereas better network

connectivities can slow down the execution time to some extent. Thus, we adjusted the timeout limit to 180 s in the $G(27,100)$ scenario, which would otherwise have more timeout results in two baseline algorithms. In contrast, the MPFRS-FC and APCRS-FC can solve all problems within 60 s. In general, the MPFRS-FC appears to have fewer timeouts, and it is more applicable in complex networks.

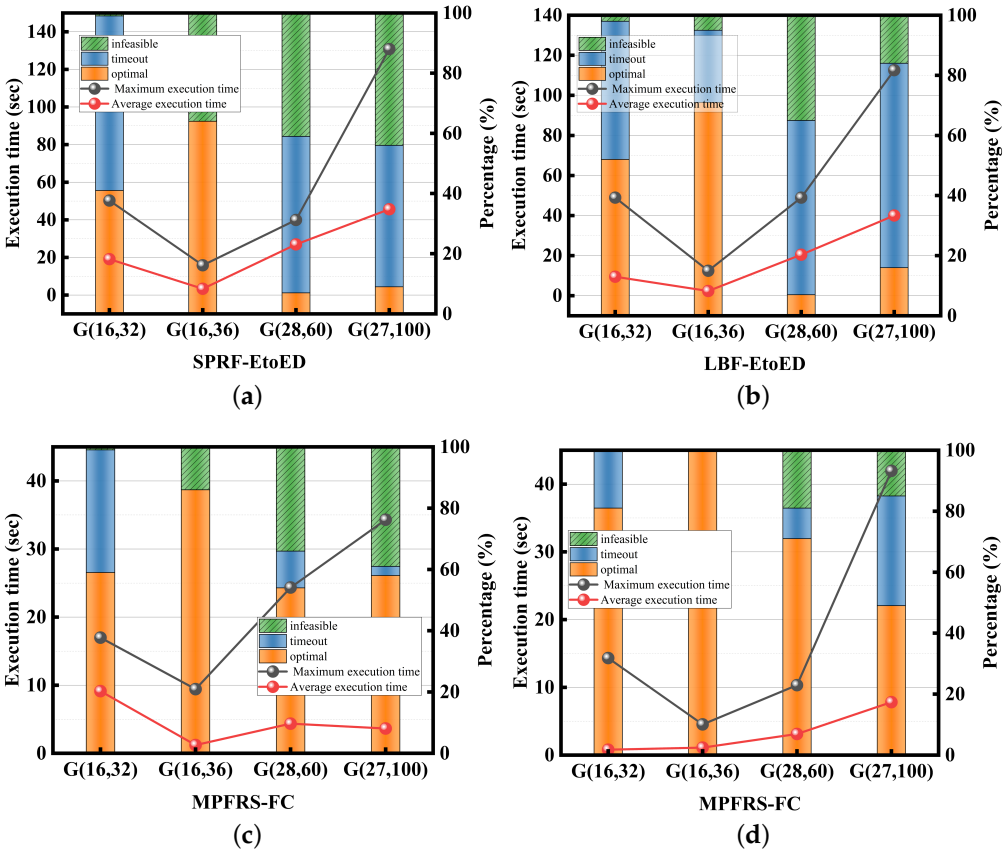


Figure 13. Percentage of solved, timeout, and infeasible schedules and execution time of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 test cases with 40 flows chosen from flow group 1, tested in MM network. (**a**) Technique 1: SPRF-EtoED. (**b**) Technique 2: LBF-EtoED. (**c**) Technique 3: MPFRS-FC. (**d**) Technique 4: APCRS-FC.

5.2.4. Performance versus Network Type

In this experiment, we compare the schedule results and the sum of end-to-end delays of the four techniques for different network types, which affects network connectivities and the options for the route. The x-axis represents the network type and its below corresponds to the algorithm shown in Figure 14. We noticed that higher network connectivities can improve the traffic schedulability, similar to the partial analysis in the previous subsection. The consequence of this is that longer routes are applied to the scheduling algorithm, resulting in increased end-to-end delays. In general, APCRS-FC obtained a higher success rate compared with other techniques in a ring network, i.e., the network topo is relatively simple and there are no two entirely nonoverlapping link routes between any two end systems.

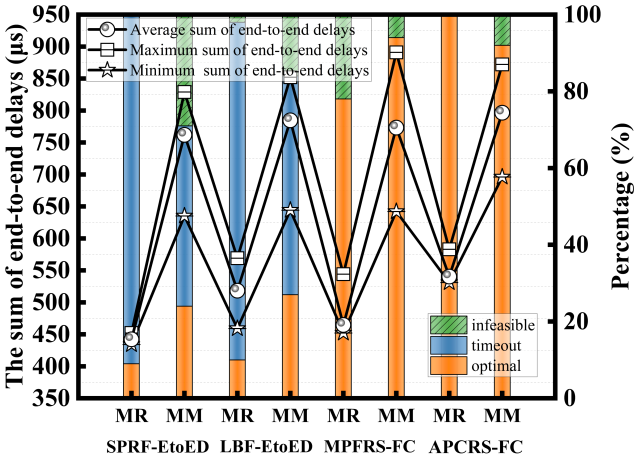


Figure 14. The sum of end-to-end delays of the SPRF-EtoED, LBF-EtoED, MPFRS-FC, and APCRS-FC for 100 test cases with 30 flows chosen from flow group 2, tested in different types of network.

6. Conclusions

In this article, the flow routing and scheduling problems in time-sensitive networking were studied. A novel method based on the FC step is proposed, which covered a finer-grained classification and the no-wait scheduling of TT flows by ILP techniques. The proposed MPFRS-FC technique covered multidisjoint path routing based on flow period correlation and links pruning to improve the scheduling performance and reduce the execution time of the algorithm. Given the characteristics of different network structures, no disjoint paths can occur in routing. An optimal scheduling method called APCRS-FC was further proposed to add an adaptive period compensation factor to make it possible for flows to increase their schedulability in the network. We designed twelve experiments, each of which used 100 test cases to evaluate the applicability scenarios and scalability of the proposed algorithm, and compared the performance with two baseline algorithms, namely, the SPRF-EtoED algorithm and LBF-EtoED algorithm. The results show the out-performance of the proposed algorithms in handling large traffic load, large differential traffic periods, large network scale, and varied network connectivity scenarios in terms of success rate and execution time. We also calculate the overall end-to-end delays in each method, though both MPFRS-FC and APCRS-FC algorithms increase the end-to-end delay of partial flows, which is not higher than 15% on average. The main reason for this depends on the routing planning, since the scheduling methods are all based on the constraints of no-wait transmission. Partial flows planned by the MPFRS-FC algorithm choose the path that is not the shortest path to the destination node to improve the global scheduling success rate. Furthermore, the APCRS-FC algorithm makes it possible to transmit in the network by compensating the traffic period at the source, which results in some flows having a delay of compensation at the first hop. This latency is accumulated in each hop relative to the original flow. Optimization here depends on a tradeoff between the latency performance of partial flows and the success rate of traffic scheduling. The larger the compensated flow period the greater the impact on the end-to-end delay performance of that flow. It is noticed that MPFRS-FC and APCRS-FC can lower the sum of end-to-end delays in large-scale networks with high traffic loads. In our future work, we would extend the routing and scheduling approach to improve scheduling flexibility by introducing deep reinforcement learning. On the other hand, the issue of converged wired and wireless routing and scheduling in TSN is also an interesting direction.

Author Contributions: Conceptualization, H.N., S.L. and Y.L.; methodology, H.N., S.L. and Y.L.; software, H.N.; validation, H.N.; formal analysis, H.N.; investigation, H.N., S.L. and Y.L.; resources, H.N., S.L., and Y.L.; data curation, H.N., S.L. and Y.L.; writing—original draft preparation, H.N.; writing—review and editing, H.N. and Y.L.; visualization, H.N.; supervision, H.N., S.L. and Y.L.; project administration, S.L.; funding acquisition, S.L. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China (No. 2020YFC1511801).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

References

1. Patzke, R. Fieldbus basics. *Comput. Stand. Int.* **1998**, *19*, 275–293. [CrossRef]
2. Bello, L.L.; Bini, E.; Patti, G. Priority-Driven Swapping-Based Scheduling of Aperiodic Real-Time Messages over EtherCAT Networks. *IEEE Trans. Ind. Inform.* **2015**, *11*, 741–751. [CrossRef]
3. Elshuber, M.; Obermaisser, R. Dependable and predictable time-triggered Ethernet networks with COTS components. *J. Syst. Archit.* **2013**, *59*, 679–690. [CrossRef]
4. Finn, N. Introduction to Time-Sensitive Networking. *IEEE Commun. Stand. Mag.* **2018**, *2*, 22–28. [CrossRef]
5. Farkas, J.; Bello, L.L.; Gunther, C. Time-Sensitive Networking Standards. *IEEE Commun. Stand. Mag.* **2018**, *2*, 20–21. [CrossRef]
6. Messenger, J.L. Time-sensitive networking: An introduction. *IEEE Commun. Stand. Mag.* **2018**, *2*, 29–33. [CrossRef]
7. Lo Bello, L.; Steiner, W. A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems. *Proc. IEEE* **2019**, *107*, 1094–1120. [CrossRef]
8. Pahlevan, M.; Obermaisser, R. Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks. In Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 4–7 September 2018.
9. Pahlevan, M.; Tabassam, N.; Obermaisser, R. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM SIGBED Rev.* **2019**, *16*, 15–20. [CrossRef]
10. Nawaz, M.; Enscore, E.E.; Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **1983**, *11*, 91–95. [CrossRef]
11. Arestova, A.; Hielscher, K.; German, R. *Design of a Hybrid Genetic Algorithm for Time-Sensitive Networking*; Springer: Cham, Switzerland, 2020.
12. 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 25th International Conference on Real-Time Networks and Systems, New York, NY, USA, 4–6 October 2017; pp. 183–192.
13. Schweissguth, E.; Danielis, P.; Timmermann, D.; Parzyjegla, H. ILP-based joint routing and scheduling for time-triggered networks. In Proceedings of the 25th International Conference on Real-Time Networks and Systems, New York, NY, USA, 4–6 October 2017; pp. 8–17.
14. Li, Q.; Li, D.; Jin, X. A Simple and Efficient Time-Sensitive Networking Traffic Scheduling Method for Industrial Scenarios. *Electronics* **2020**, *9*, 2131. [CrossRef]
15. Nayak, N.G.; Duerr, F.; Rothermel, K. Routing algorithms for IEEE802.1Qbv networks. *SIGBED Rev.* **2018**, *15*, 13–18. [CrossRef]
16. Syed, A.A.; Ayaz, S.; Leinmüller, T.; Chandra, M. MIP-based Joint Scheduling and Routing with Load Balancing for TSN based In-vehicle Networks. In Proceedings of the 2020 IEEE Vehicular Networking Conference (VNC), New York, NY, USA, 16–18 December 2020; pp. 1–7.
17. Yu, Q.; Gu, M. Adaptive Group Routing an Scheduling in Multicast Time-Sensitive Networks. *IEEE Access* **2020**, *8*, 37855–37865. [CrossRef]
18. Li, C.; Zhang, C.; Zheng, W.; Wen, X.; Lu, Z.; Zhao, J. Joint Routing and Scheduling for Dynamic Applications in Multicast Time-Sensitive Networks. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
19. Craciunas, S.S.; Oliver, R.S. An overview of scheduling mechanisms for time-sensitive networks. In *Real-Time Summer School L'École d'Été Temps Réel (ETR)*; Technical Report; TTTech Computertechnik AG: Vienna, Austria, 2017; pp. 1–7.
20. Dürr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212.

21. IEEE, 802.1 Qbv. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic. In IEEE Std 802.1Qbv-2015, 18 March 2016, pp. 1–57. Available online: https://ieeexplore.ieee.org/Xplore/home.jsp (accessed on 5 April 2022).
22. Gavrilut, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 13–15 June 2018; pp. 1–4.
23. Pop, P.; Raagaard, M.L.; Craciunas, S.S.; Steiner, F. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Phys. Syst. Theory Appl.* **2016**, *1*, 86–94. [CrossRef]
24. Craciunas, S.S.; Oliver, R.S. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Syst.* **2016**, *52*, 161–200. [CrossRef]
25. Serna Oliver, R.; 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; pp. 13–24.
26. Shalghum, K.; Noordin, N.K.; Sali, A.; Hashim, F. Network Calculus-Based Latency for Time-Triggered Traffic under Flexible Window-Overlapping Scheduling (FWOS) in a Time-Sensitive Network (TSN). *Appl. Sci.* **2021**, *11*, 3896. [CrossRef]
27. Nayak, N.G.; Durr, F.; Rothermel, K. Incremental flow scheduling and routing in time-sensitive software-defined networks. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2066–2075. [CrossRef]
28. Laursen, S.M.; Pop, P.; Steiner, W. Routing optimization of AVB streams in TSN networks. *SIGBED Rev.* **2016**, *13*, 43–48. [CrossRef]
29. Kim, H.-J.; Lee, K.-C.; Kim, M.-H.; Lee, S. Optimal Scheduling of Time-Sensitive Networks for Automotive Ethernet Based on Genetic Algorithm. *Electronics* **2022**, *11*, 926. [CrossRef]
30. Gavrilut, V.; Zhao, L.; Raagaard, M.L.; Pop, P. AVB-aware routing and scheduling of time-triggered traffic for TSN. *IEEE Access* **2018**, *6*, 75229–75243. [CrossRef]
31. 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; pp. 375–384.
32. Hanzalek, Z.; Burget, P.; Sucha, P. Profinet IO IRT message scheduling with temporal constraints. *IEEE Trans. Ind. Inform.* **2010**, *6*, 369–380. [CrossRef]
33. Scholer, C.; Krenz-Baath, R.; Murshed, A.; Obermaisser, R. Computing optimal communication schedules for time-triggered networks using an SMT solver. In Proceedings of the 2016 11th IEEE Symposium on Industrial Embedded Systems (SIES), Krakow, Poland, 23–25 May 2016; pp. 83–91.
34. Feng, T.; Yang, H. SMT-based Task- and Network-level Static Schedule for Time Sensitive Network. In Proceedings of the 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), Kuala Lumpur, Malaysia, 14–16 May 2021; pp. 764–770.
35. Jin, X.; Xia, C.; Guan, N.; Zeng, P. Joint algorithm of message fragmentation and no-wait scheduling for time-sensitive networks. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 478–490. [CrossRef]
36. Atallah, A.A.; Hamad, G.B.; Mohamed, O.A. Routing and Scheduling of Time-Triggered Traffic in Time-Sensitive Networks. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4525–4534. [CrossRef]
37. Xu, L.; Xu, Q.; Zhang, Y.; Zhang, J.; Chen, C. Co-design approach of scheduling and routing in time sensitive networking. In Proceedings of the 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Tampere, Finland, 10–12 June 2020; pp. 111–116.
38. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 27: Enhancements to Bridging of IEEE 802.11 Media. In IEEE Std 802.1Qbz-2016 (Amendment to IEEE Std 802.1Q-2014), 30 September 2016; pp. 1–26. Available online: https://ieeexplore.ieee.org/Xplore/home.jsp (accessed on 5 April 2022).
39. Arif, F.A.R.; Atia, T.S. Load balancing routing in time-sensitive networks. In Proceedings of the 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, Ukraine, 4–6 October 2016; pp. 207–208.
40. Ojewale, M.A.; Yomsi, P.M. Routing heuristics for load-balanced transmission in TSN-based networks. *SIGBED Rev.* **2019**, *16*, 20–25. [CrossRef]