electronics

MDPI

Article

Traffic Classification and Packet Scheduling Strategy with Deadline Constraints for Input-Queued Switches in Time-Sensitive Networking

Ling Zheng *, Guodong Wei, Keyao Zhang and Hongyun Chu

School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710061, China; guodongwei@stu.xupt.edu.cn (G.W.); keyaoz@stu.xupt.edu.cn (K.Z.); hychu@xupt.edu.cn (H.C.)

* Correspondence: lingzheng@xupt.edu.cn

Abstract: Deterministic transmission technology is a core key technology that supports deterministic real-time transmission requirements for industrial control in Time-Sensitive Networking (TSN). It requires each network node to have a deterministic forwarding delay to ensure the real-time end-to-end transmission of critical traffic streams. Therefore, when forwarding data frames, the switch nodes must consider the time-limited requirements of the traffic. In the input-queued switch system, an algorithm for clock-synchronized deterministic network traffic classification scheduling (CSDN-TCS) is proposed to address the issue of whether a higher-quality-of-service (QoS) performance can be provided under packet deadline constraints. First, the scheduling problem of the switch is transformed into a decomposition problem of the traffic matrix. Secondly, the maximum weight-matching algorithm in graph theory is used to solve the matching results slot by slot. By fully utilizing the slot resources, as many packets as possible can be scheduled to be completed before the deadline arrives. For two types of packet scheduling problems, this paper uses the maximum flow algorithm with upper- and lower-bound constraints to move packets from a larger deadline set to idle slots in a smaller deadline set, enabling early transmission, reducing the average packet delay, and increasing system throughput. When there are three or more types of deadlines in the scheduling set, this scheduling problem is an NP-hard problem. We solve this problem by polling the two types of scheduling algorithms. In this paper, simulation experiments based on the switching size and line load are designed, and the Earliest Deadline First (EDF) algorithm and the Flow-Based Iterative Packet Scheduling (FIPS) algorithm are compared with the CSDN-TCS algorithm. The simulation results show that under the same conditions, the CSDN-TCS algorithm proposed in this paper outperforms the other two algorithms in terms of success rate, packet loss rate, average delay and throughput rate. Compared with the FIPS algorithm, the CSDN-TCS algorithm has lower time complexity under the same QoS performance.

Keywords: time-sensitive networking; input-queuing system; multi-class packet scheduling; deadline guarantee; quality of service; NP-hard

check for updates

Citation: Zheng, L.; Wei, G.; Zhang, K.; Chu, H. Traffic Classification and Packet Scheduling Strategy with Deadline Constraints for Input-Queued Switches in Time-Sensitive Networking. *Electronics* **2024**, *13*, 629. https://doi.org/10.3390/electronics13030629

Academic Editor: Christos J. Bouras

Received: 27 December 2023
Revised: 24 January 2024
Accepted: 26 January 2024
Published: 2 February 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

Faced with the challenge of network congestion and packet delay resulting from the rapid surge in audio/video traffic and the proliferation of real-time industrial machine applications, there is a growing demand for the real-time transmission and processing of large-scale high-speed data across various domains. For instance, the need for swift adaptation to environmental shifts in autonomous driving, instantaneous data upload and control command delivery in telemedicine, and real-time information exchange in virtual reality systems necessitates the control of delays within the range of 1 to 10 milliseconds, with jitter control at the microsecond level [1]. Traditional Ethernet, due to the absence of a clock synchronization mechanism, bandwidth reservation management mechanism, data

packet priority, and other filtering mechanisms, as well as the incompatibility between different protocols, can only mitigate end-to-end delay to the order of tens of milliseconds [2]. Consequently, it fails to provide the requisite quality of service (QoS) assurance for delay and jitter, and falls short of meeting the real-time deterministic transmission requirements for these applications [3].

In November 2012, the IEEE 802.1 working group extended and upgraded Audio Video Bridging (AVB) to the Time-Sensitive Networking (TSN) working group, proposing a series of standards and specifications for link-layer enhancements and traffic policies. These primarily include standards for time synchronization, traffic scheduling, reliable transmission, and network management, aimed at extending the standard Ethernet 802.3 to support real-time and deterministic data transmission for industrial control [4]. TSN adheres to the standard Ethernet protocol suite, naturally possessing superior interoperability advantages. It can provide deterministic latency, bandwidth guarantees, and achieve open 2-layer forwarding [5]. TSN ensures the real-time and reliable transmission of data through a series of key technologies, such as high-precision time synchronization, resource reservation and path control, traffic shaping, and traffic matrix scheduling, thereby guaranteeing the quality of service for time-critical and event-critical applications [6]. In deterministic transmission, packets have known and deterministic delays and are transmitted and received in a predictable manner [7].

The currently released TSN specifications primarily cover four key areas: clock synchronization, data latency, reliability, and resource management. Among these, clock synchronization is considered the foundation of TSN technology. The IEEE 802.1AS standard precisely defines a high-precision clock synchronization mechanism, enabling devices in the network to achieve synchronization at the microsecond level. This ensures that end devices and network equipment can operate in coordination according to the same clock. In addressing scheduling and queuing mechanisms, the IEEE 802.1Qbv standard focuses on the functionality of a time-aware shaper in a strict priority mode with multiple output queues in switches. Through the use of a gate control list, this standard effectively controls the switching time window for each queue, facilitating efficient time-sensitive communication. The IEEE 802.1Qbu standard allows high-priority data to preempt low-priority data, ensuring the real-time transmission of critical control information. Regarding reliability, the IEEE 802.1Qca and IEEE 802.1Qci standards are dedicated to enhancing the reliability level of TSN networks. Additionally, the IEEE 802.1Qat and IEEE 802.1Qcc standards focus on resource management in TSN, including network topology configuration and traffic engineering specifications. These standards collectively establish a unified framework, laying the foundation for the widespread application of TSN technology. The continuous development of TSN and the ongoing updates to standards will further drive advancements in real-time communication technology, providing industries with a more efficient and reliable network infrastructure.

Traffic scheduling is a core mechanism in the TSN standard, utilizing scheduling algorithms to determine the order and timing of data frame transmission at all switch output ports. This ensures compliance with the individual latency and bandwidth requirements of each traffic flow while optimizing the overall transmission performance [8]. TSN categorizes transmitted data into three main types based on different traffic requirements: time-triggered (TT) traffic, rate-constrained (RC) real-time traffic, and best effort (BE) non-real-time traffic [9]. Each type of traffic imposes different demands on end-to-end latency and bandwidth within the network. Traffic scheduling, achieved through the use of scheduling algorithms, produces a scheduling table. This table is then utilized to ascertain the transmission order and timing of each data frame at the output ports of a switch. The goal is to ensure the sequential transmission of data frames on the output link without conflicts, meeting the latency and bandwidth requirements of each flow. This facilitates the coexistence of diverse traffic flows on a single network [5]. To meet elevated QoS assurances, the selection of suitable switching systems and the proposal of more efficient switching scheduling algorithms become imperative [10].

The commonly used high-performance packet-switching systems include output queuing systems, input queuing systems, buffered switching systems and so on [11]. Output queuing systems can control packet delay and provide QoS guarantees [12–14]. However, in an N-port switching system, the exchange structure needs to schedule packets at a rate N times that of the input port link, which may lead to resource wastage and poor scalability. On the other hand, input-queuing systems effectively avoid scalability issues. In an input-queuing switching system, packets first queue at the input port and then undergo transmission scheduling, largely avoiding competition at the output ports. However, the single first-in, first-out (FIFO) queue at the input port leads to head-of-line (HOL) blocking, resulting in a decrease in the switch's throughput. Researchers have addressed the HOL problem by employing virtual output queuing at the input port [15]. Nevertheless, input-queuing switches based on virtual output queuing still use a first-in, first-out forwarding method and do not consider the time-sensitive nature of traffic, thus failing to guarantee forwarding delays [16,17].

The packet switching with deadline guarantees plays a crucial role in ensuring the timely transmission of critical data packets. This technological framework enables networks to efficiently safeguard the quality-of-service performance of real-time applications, preventing a degradation in quality caused by delays in real-time traffic. Therefore, in-depth research into packet scheduling with deadline guarantees is imperative to ensure that networks can maintain the stable and reliable real-time transmission of packets, especially in environments with large switch sizes and high loads. Such research not only advances real-time communication technology but also provides industries with a more efficient and reliable network infrastructure, contributing to the innovation and development of future networks.

In order to achieve deterministic transmission, this paper transforms the packet-switching scheduling problem into a traffic matrix decomposition problem. It sets transmission deadlines, for a set of packets [18]. In the input-queuing switching system, the aim of this paper is to find an efficient scheduling algorithm that allows as many packets as possible to be transmitted before the deadline arrives [19,20].

The scheduling set contains packets with the same deadline, which are classified into the same category, while packets with different deadline constraints should be classified into different categories. Therefore, the deadline-constrained packet scheduling problem can be divided into single-class packet scheduling, two-class packet scheduling, and multi-class packet scheduling based on the number of deadlines [21]. The von Neumann matrix decomposition algorithm is directly used to decompose the traffic matrix to solve the single-class scheduling problem in reference [22]. Unfortunately, the limitation of the von Neumann matrix decomposition algorithm is that it is only applicable to solving single-class packet scheduling problems and cannot effectively solve two-class or multi-class scheduling problems. Traditional methods for solving the multi-class scheduling problem involve using algorithms such as Earliest Deadline First (EDF) or Minimum Laxity First (MLF) and their variants as detailed in [23–26]. Traditional heuristic algorithms have the advantage of low complexity, but they also have room for improvement in QoS performance metrics, such as throughput, packet loss, and delay. Ref. [27] proposes the iSLIP algorithm based on highest priority scheduling, which reduces the delay at the input end, but its performance is only average in non-uniform traffic scenarios. Ref. [15] proposes a more efficient algorithm, different from traditional heuristic algorithms, such as EDF, called the Flow-Based Iterative Packet Scheduling (FIPS) algorithm. This algorithm achieves advanced scheduling by employing a maximum flow algorithm with upper and lower bounds, ensuring that packets with larger deadlines are scheduled ahead into the available time slots of packets with smaller deadlines. It improves the QoS performance to a certain extent and has been tested on large switch sizes. However, the FIPS algorithm has a high time complexity. Additionally, the FIPS algorithm requires packets in input and output ports to satisfy the scheduling conditions in two-class packet scheduling in order to complete the scheduling. In other words, as long as the traffic matrix does not satisfy the scheduling conditions,

packet scheduling cannot be performed. Clearly, the FIPS algorithm places high demands on the traffic matrices in the scheduling set. Ref. [28] proposes a nested periodic flow optimization scheduling algorithm, which decomposes multi-class scheduling problems into two-class and single-class scheduling problems through dimensionality reduction and uses an optimization solver to obtain the scheduling results. Refs. [29–31] apply a genetic algorithm to input-queuing switching systems to solve the maximum flow in group scheduling, completing simulations with high throughput, but the article only tested it on a small switch size and has not yet verified its algorithm performance in multi-port switching scenarios. Recently, Ref. [32] proposed an exhaustive priority service empty queue and hybrid weight algorithm, achieving high throughput and low latency under Bernoulli uniform traffic. However, when faced with bursty large traffic and multi-port scenarios, there is still room for improvement in its algorithm's QoS performance.

Based on the research and analysis of the above algorithms, this paper proposes an algorithm, referred to as the Clock Synchronization Deterministic Network Traffic Classification Scheduling algorithm (CSDN-TCS). In single-class scheduling, this algorithm utilizes the maximum weight-matching algorithm on a per-time-slot basis to decompose the traffic matrix [33]. It ensures that within each time slot before the deadline, each input port group can transmit the maximum number of packets to the output port without encountering competition, thereby fully utilizing time slot resources to accomplish the packet scheduling task.

In the two-class packet scheduling, the CSDN-TCS algorithm employs a maximum flow algorithm with upper and lower bounds to determine the maximum legal flow. It schedules some packets with larger deadlines ahead into the available time slots of packets with smaller deadlines. The key distinction between the CSDN-TCS algorithm and the FIPS algorithm lies in the fact that, when the FIPS algorithm satisfies the scheduling conditions for the larger deadline traffic matrix—specifically, when there is no excessive packet occurrence at each port—it ceases the advanced scheduling of packets, proceeding directly to the decomposition of the traffic matrix. If there are still idle slots in the scheduling time slots of packets with smaller deadlines, these idle slot resources cannot be fully utilized. It means that there is still room for improvement in the QoS performance of the algorithm. The proposed CSDN-TCS algorithm is based on the idea that, as long as there are available time slot resources at the input and output ports of the traffic matrix with smaller deadlines, it iteratively extracts some packets with larger deadlines. It schedules them in advance in the available time slots of packets with smaller deadlines until there are no more available time slot resources in the traffic matrix with smaller deadlines or no more packets that can be scheduled in advance from the traffic matrix with larger deadlines. The algorithm terminates under these conditions. This approach serves to enhance the throughput of the switching system and reduce the average packet delay. Simulation results show that compared with traditional EDF algorithms and FIPS algorithms, the CSDN-TCS algorithm can solve the packet-scheduling problem with deadline guarantees with lower packet loss, higher throughput, and smaller average delay. Moreover, under comparable QoS performance, the time complexity of the CSDN-TCS algorithm is lower and easier to implement. The CSDN-TCS algorithm does not impose additional restrictions on the generation of traffic matrices in two-class packet scheduling, making it more adaptable to general scheduling situations and more in line with actual scheduling scenarios.

The contributions of this paper are as follows:

Firstly, this paper establishes a model for the input-queuing switch system in TSN. It further delves into the operational mechanism of the input queuing system and the scheduling process of packets in TSN networks. On this basis, it clarifies the necessity of studying packet scheduling with deadline guarantees.

Secondly, the paper proposes the CSDN-TCS algorithm, which aims to address how to schedule as many packets as possible under deadline constraints. This algorithm effectively improves the throughput of the switching system, ensuring a reduction in the average

delay of packets while maintaining a low packet loss rate, thus enhancing the efficiency of the system.

Finally, we conduct a series of simulation experiments from the perspectives of switch size and line load. We verify the superior QoS performance of the CSDN-TCS algorithm in terms of success rate, packet loss rate, average delay, and throughput.

The rest of this article is organized as follows. The Section 2 introduces the system model and preparation of the packet scheduling problem with deadline guarantee. The Section 3 introduces the design of the algorithm. The Section 4 gives the simulation results and comparative analysis. The Section 5 summarizes the full text.

2. System Model and Preparation

2.1. System Model

This paper establishes a structural model for the TSN input–queuing switch system as shown in Figure 1. In this model, the TSN switch has N input ports and N output ports. Firstly, arriving packets at the input ports are queued, and after passing through virtual output queues, they enter the switching engine. Secondly, the switching system utilizes the IEEE 802.1Qci and IEEE 802.1Qcr standards for packet filtering and asynchronous traffic shaping. The packets enter queues with different priorities. Subsequently, these queues, under the control of a gate control list, open time windows and transmit corresponding traffic flows. Finally, after priority selection, the packets reach the output ports.

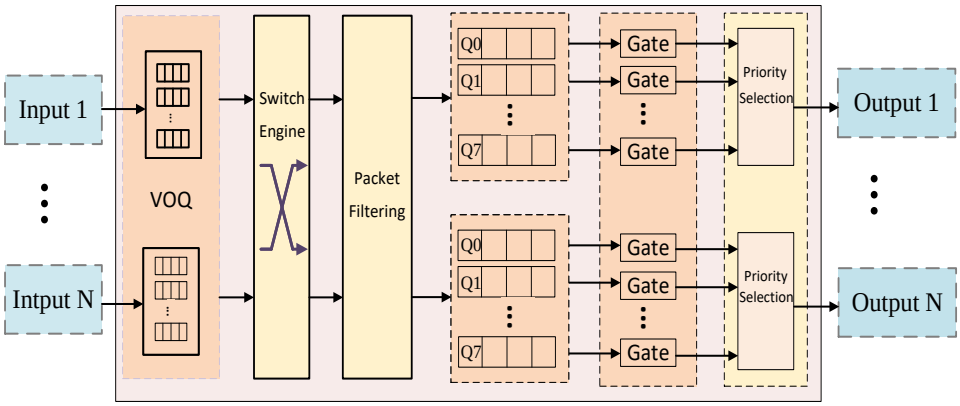


Figure 1. TSN input-queuing switch system model.

In the model, the gate control array directly acts on the positions of the output ports. If packets arriving at the input ports are not forwarded to the output ports before the deadline, the gate control array cannot exert control. Therefore, studying the scheduling of flows with deadline guarantees is essential in this context.

The commonly used symbols and variables along with their meanings in this paper are provided in Table 1.

This paper abstracts the input port and output port of the switch into a bipartite graph model as shown in Figure 2, which divides the input port and output port of the switch into two disjoint subsets U,V, let $G_t = (V, E)$, where the set of input ports is U, $U = \{u_1, u_2, \ldots, u_n\}$, and the set of output ports is V, $V = \{v_1, v_2, \ldots, v_n\}$. Each port can only belong to one subset. Packets arriving at an input port will be directed to the corresponding output port. The bipartite graph-matching method matches the input ports of the switch to unique output ports, enabling conflict-free packet transmission and exchange, thereby improving the switching efficiency and enhancing the throughput of the switching system. In network switches, the use of bipartite graph models can provide a deep understanding of the network topology and help optimize network performance.

Table 1. Summary of notations.

Notation	Description
G_t	The bipartite graph
U	The set of input ports
V	The set of output ports
E	The edges in a bipartite graph
S	The set of total packet schedules
u_i	The input port of the switch
v_j	The output port of the switch
D	The set of deadlines
p_i^d	The total number of packets in the input port
q_j^d	The total number of packets in the output port
a_i	The excess of input port i of the traffic matrix
b_j	The excess of output port j of the traffic matrix
c_i	The available capacity of input port i of the traffic matrix
e_j	The available capacity of output port j of the traffic matrix
$p_{i,j}(t)$	The total number of packets in the input port
$p_i^{d_l}$	The number of packets at the input port
$q_j^{d_l}$	The number of packets at the output port



(**a**) (**b**)

Figure 2. Bipartite graph model. (**a**) Bipartite graph. (**b**) Bipartite graph matching.

This paper transforms the packet scheduling problem of the switch into a traffic matrix decomposition problem by placing packets arriving at various input ports of the switch into the scheduling set S. Based on parameters such as input port, destination port, deadline, and packet quantity of the packets in S, a traffic matrix is created. An example is given with $N = 4$ and $d = 3$ as shown in Figure 3.

$$A = \begin{array}{c|cccc|c} & v_1 & v_2 & v_3 & v_4 & p_i^d \\ \hline u_1 & 1 & 0 & 3 & 0 & 4 \\ u_2 & 1 & 1 & 0 & 1 & 3 \\ u_3 & 3 & 0 & 2 & 1 & 6 \\ u_4 & 1 & 1 & 0 & 0 & 2 \\ \hline q_j^d & 6 & 2 & 5 & 2 & \end{array}$$

Figure 3. Class A traffic matrix.

In the traffic matrix, u_i $(1 \leqslant i \leqslant N)$ indicates switch input ports and v_j $(1 \leqslant j \leqslant N)$ indicates switch output ports. p_i^d $(1 \leqslant i \leqslant N)$ represents the total number of packets in the input port, q_j^d $(1 \leqslant j \leqslant N)$ represents the total number of packets in the output port,

and the superscript d represents the deadline for class A packets. The element coordinates are represented as (i, j), where the abscissa i represents the i-th input port in the switch, and the ordinate j represents the j-th output port in the switch. The numerical value of each element indicates the number of packets sent from input port i to output port j. For example, element size 2 at position (3, 3) in matrix A in Figure 3 indicates that the switch needs to transmit 2 packets from input port 3 to output port 3 before the deadline d.

Definition 1. *The excess of input port i in the traffic matrix is denoted as* a_i, $a_i = p_i^{d_1} - (d_1 + 1)$. *The excess of output port is* b_j, $b_j = q_j^{d_1} - (d_1 + 1)$. *If* $a_i > 0$ *or* $b_j > 0$, *it indicates that the input or output ports of the traffic matrix are overloaded. If* $a_i < 0$ *or* $b_j < 0$, *it represents that the ports of the service matrix are still vacant.*

Definition 2. *The available capacity of input port i in the traffic matrix is denoted as* c_i, $c_i = (d_1 + 1) - p_i^{d_1}$. *The availability of the output port is* e_j, $e_j = (d_1 + 1) - q_j^{d_1}$. *If* $c_i > 0$ *or* $e_j > 0$, *it indicates that the input ports or output ports of the traffic matrix are vacant; otherwise, if* $c_i < 0$ *or* $e_j < 0$, *it indicates that the input ports and output ports of the traffic matrix are overloaded.*

The available capacity of the time slots in $[0, d_1]$ and the excess of time slots in $[d_1 + 1, d_2]$ can be calculated through the traffic matrix.

2.2. Preparation and Question Formula

Based on the TSN input queuing system exchange model mentioned above, the goal of researching the deadline-guaranteed differentiated services scheduling algorithm is to design a scheduling algorithm that maximizes the network throughput based on the traffic matrix of incoming traffic. Under the constraint of deadlines, the aim is to schedule as many packets as possible. To accomplish this scheduling task, we make the following assumptions, which serve as the foundation for designing and implementing the algorithm.

Assumption 1. *In an input-queued switch, the time axis is divided into equally spaced time slots, with the length of each slot being equal to the time required for the transmission or reception of a single packet. The time slots are numbered starting from 0.*

Assumption 2. *In each time slot, the switch establishes a one-to-one connection from an input port to an output port. Each input port or output port can send or receive at most one packet in a time slot, and packets can only be sent or received at the beginning of a time slot. It is assumed that all packets are of a fixed size.*

Assumption 3. *Let each packet be associated with a triple* (i, j, d_n), *where the* d_n *indicates the deadline of packets, let* $D = \{d_1, d_2, d_3, \ldots, d_n\}$, $(d_1 < d_2 < d_3 < \ldots < d_n)$. *The presence of deadlines guarantees that packets must be transmitted from input port i to output port j before the deadline. Packets that miss the deadline will be discarded [34].*

Assumption 4. *Packets with smaller deadlines are assigned higher priority, while those with larger deadlines have lower priority. The switching system will prioritize the scheduling of packets with smaller deadlines, followed by those with larger deadlines, ultimately completing the scheduling of all packets [35].*

Theorem 1. *For a packet set S, its corresponding set of deadline* $D = \{d_1, d_2, d_3, \ldots, d_n\}$, $(d_1 < d_2 < d_3, \ldots, < d_n)$. *When the following conditions are met, the set can be referred to as non-overloaded:*

$$\sum_{l=1}^{p} p_i^{dl} \leqslant d_p + 1 \tag{1}$$

$$\sum_{l=1}^{p} q_j^{dl} \leqslant d_p + 1 \tag{2}$$

$$\forall i, j \in \{1, 2, \ldots N\}\ \forall p \in \{1, 2, \ldots n\}$$

Theorem 1 is easily proven because the $\sum_{l=1}^{m} p_i^{dl}$ represents the total number of packets arriving at the i-th input port, $\sum_{l=1}^{m} q_j^{dl}$ represents the total number of packets arriving at the j-th output port, and $d_m + 1$ represents the total number of available time slots. When the total number of arriving packets at an input or output port is less than or equal to the total number of available time slots, the system load does not exceed 1, and the system is considered non-overloaded. Otherwise, the system is in an overloaded state.

Based on the analysis of the research objectives, assumptions, and theorems mentioned above, the problem of scheduling classified traffic with deadline guarantees in TSN input queuing switch systems can be described as follows:

$$\max \sum_{t=1}^{d_m} \sum_{i=1}^{N} \sum_{j=1}^{N} |p_{i,j}(t)| \tag{3}$$

Due to the fact that, at most, one packet can be transmitted between an input port and an output port in any time slot, the following constraints must be satisfied for each time slot:

$$|p_{i,j}(t)| = \{0, 1\} \quad \forall i, j \in \{1, \ldots N\} \tag{4}$$

$$\sum_{j=1}^{N} p_{i,j}\ (t) \leqslant 1 \quad \forall i \in \{1, \ldots N\} \tag{5}$$

$$\sum_{i=1}^{N} p_{i,j}\ (t) \leqslant 1 \quad \forall j \in \{1, \ldots N\} \tag{6}$$

where $|p_{i,j}(t)|$ represents the total number of packets scheduled in slot t.

This paper first models and analyzes the scheduling problem, and identifies the main approach to solving it as follows: Initially, establish an input traffic matrix based on parameters such as input port, destination port, deadline, and number of packets for the input traffic. Then, based on the number of deadlines for the groups in the scheduling set, discuss separately. If all groups in the scheduling set have the same deadline, the maximum weight matching algorithm can be employed to decompose and solve the input traffic matrix. If the groups in the scheduling set have two or more different deadlines, this paper proposes to establish a network flow model based on the input traffic matrix. The scheduling problem is then transformed into a maximum flow problem with upper and lower bounds on the flow, and solved using graph theory's maximum flow algorithms. This approach is designed to create a scheduling algorithm that satisfies the deadline constraints and maximizes the throughput.

3. Algorithm Design

Grouping packets with the same deadline from the scheduling set into a specific collection, establish a corresponding traffic matrix for each group of packets. A network flow G is created based on the traffic matrix, then a maximum flow algorithm is applied to determine the maximum network flow, which corresponds to an optimal scheduling. Utilize a maximum weight matching algorithm in each time slot to decompose the traffic matrix, ensuring that in each time slot, each input port can only send data to one output port, and simultaneously, each output port can only receive data from one input port. Obtain the final scheduling result through this process. The algorithm can be broadly categorized into three scenarios, each of which will be discussed in the following sections.

3.1. Single-Class Packet Scheduling

Single-class packet scheduling refers to a scheduling set in which all packets have the same deadline. The algorithm proposed in this paper achieves maximum flow by iteratively invoking the maximum weight matching algorithm within the [0, deadline] range. The system's throughput is enhanced, and time slot resources are fully utilized, allowing as many packets as possible to be scheduled while ensuring deadline guarantees. The following introduces the core algorithm of single-class packet scheduling—the maximum weight matching algorithm.

Maximum Weight Matching Algorithm

The maximum weight matching algorithm, as shown in the Algorithm 1, is used to find a matching in a bipartite graph, where each edge in the bipartite graph is associated with a vertex, and this matching has the maximum sum of weights.

Algorithm 1 Maximum weight matching algorithm.

(1) Initialize the top label and assign a value to each vertex;
(2) Use Hungarian algorithm to find the perfect match;
(3) If no perfect match is found, modify the top marker value;
(4) Repeat steps (2) and (3) until you find a perfect match for the equivalent subgraph;

The time complexity of the maximum weight matching algorithm is $O(N^3)$.

Because the time complexity of the single-class packet scheduling algorithm primarily depends on the maximum weight matching algorithm, the complexity of the single-class packet scheduling algorithm, as shown in the Algorithm 2, is also $O(N^3)$.

Algorithm 2 Single-class packet scheduling algorithm.

Input: A
Output: M_i
1: **for** $d = 0 : d_1$ **do** $//d$ represents the sequence of time slot
2: Called maximum weight matching algorithm(A)
3: Output matching matrix M_i
4: Update the traffic matrix A
5: **end for**

3.2. Two Types of Packet Scheduling

If there are two different deadlines $D = \{d_1, d_2\}$ in the total packet set and they are met $0 < d_1 < d_2$, put the packets whose deadline is d_1 into set A, and the packets whose deadline is d_2 into B, and let $S = AUB$; this situation is referred to as two-class packet scheduling. The approach of the two-class scheduling algorithm is to prioritize the scheduling of packet sets with smaller deadlines. If there are idle time slots during the scheduling process for the packet sets with smaller deadlines, some of the packets with larger deadline can be scheduled in advance to occupy those time slots.

Advance Scheduling Algorithm

The advance scheduling algorithm, as shown in the Algorithm 3, is shown below, and its time complexity is $O(n^2 \times m)$, where n is the number of nodes and m is the number of edges.

As an example shown in Figure 4, suppose there are two 4×4 traffic matrices and the deadline of A class is $d_1 = 9$. The deadline of B class is $d_2 = 11$.

$A=$

	v_1	v_2	v_3	v_4	p_i^2	c_i
u_1	0	1	2	0	3	7
u_2	2	1	0	1	4	6
u_3	2	1	2	2	7	3
u_4	1	1	1	2	5	5
q_j^2	5	4	5	5		
e_j	5	6	5	5		

$B=$

	v_1	v_2	v_3	v_4	p_i^2	a_i
u_1	2	2	1	2	7	5
u_2	1	1	2	2	6	4
u_3	1	1	2	1	5	3
u_4	0	2	1	1	4	2
q_j^2	4	6	6	6		
b_j	2	4	4	4		

Figure 4. Traffic matrices A and B.

Algorithm 3 Advance (A, B).

Input: Traffic matrix A, B
Output: Updated traffic matrix A, B

(1) Calculate the emptiness of A. $c_i = (d_1 + 1) - p_{ij}^{d_1}$, $e_j = (d_1 + 1) - q_{ij}^{d_1}$.
(2) Calculate the excess of B. $a_i = p_{ij}^{d_2} - (d_2 - d_1)$, $b_j = q_{ij}^{d_2} - (d_2 - d_1)$.
(3) Construct a four-level flow network with five parameters $G(s, t, U, V, E)$, where the source point s is the first level of the flow network. The second-level flow network has a set U consisting of N nodes, $U = \{u_1, u_2, \ldots, u_n\}$ and u_i $(1 \leqslant i \leqslant N)$ represents the i-th input port. There also exists a set V containing N nodes in the third-level flow network, $V = \{v_1, v_2, \ldots, v_n\}$, where v_j $(1 \leqslant i \leqslant N)$ represents the j-th output port. The sink node t belongs to the fourth level of the flow network. The set E represents an edge between two nodes, and each edge in the set is assigned a lower boundary and an upper boundary.
(4) Connect s and u_i $(1 \leqslant i \leqslant N)$, when the condition $0 < a_i < c_i$ is met, the lower bound $l_{si} = \max\{0, a_i\}$, otherwise, $l_{s_i} = 0$. The upper bound $u_{s_i} = c_i$. The purpose of setting upper and lower bounds is to ensure that the number of packets scheduled from input port i of matrix B to input port of matrix A does not exceed the available capacity c_i of matrix A, while ensuring that the input ports of matrix B do not become overloaded as much as possible.
(5) Join the edge between u_i and v_j $(1 \leqslant i, j \leqslant N)$. Then, set the lower bound $l_{ij} = 0$, upper bound $u_{ij} = p_{ij}^{d_2}$, which means that we can select some of packets from the B set, and $p_{ij}^{d_2}$ to be scheduled in the free time slot of A set ahead of the deadline.
(6) Connect v_j $(1 \leqslant j \leqslant N)$ and t when $0 < b_j < e_j$, lower bound $l_{jt} = \max\{0, b_j\}$; otherwise, $l_{jt} = 0$, and upper bound $u_{jt} = e_j$, which means that, on the port of B, at most, e_j packets will be promoted to the A set along with the packets for transmission scheduling.
(7) Call the upper and lower bound maximum flow algorithm to find the maximum feasible flow *maxflow*.
(8) Output advance matrix B_x ($the B_x$ matrix represents a set of packets scheduled from the B matrix advance to the A matrix's free time slot);
(9) Update the traffic matrix A: $A = A + B_x$. The updated A set includes the original A set and the packets from B class in advance;
(10) Update the traffic matrix B: $B = B - B_x$. The updated B set includes the original B set minus the pre-scheduled partial grouping.
(11) End

Figure 5 is a flow network constructed by the traffic matrices A and B, and Figure 6 is a maximum legitimate flow in the flow network calculated by the maximum flow algorithm, which means that 13 packets will be promoted from the category B set and scheduled in advance together with the category A packets in their free time slots.

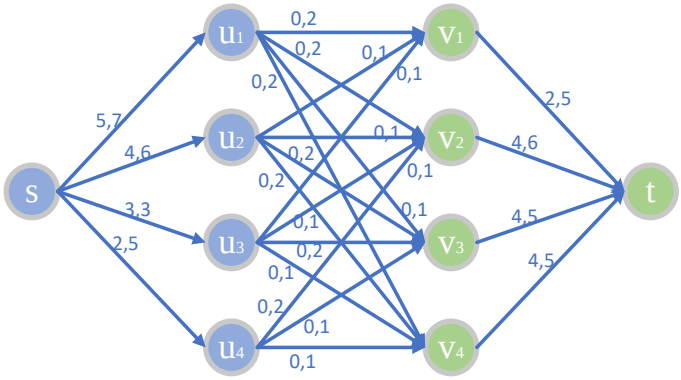


Figure 5. Stream network.

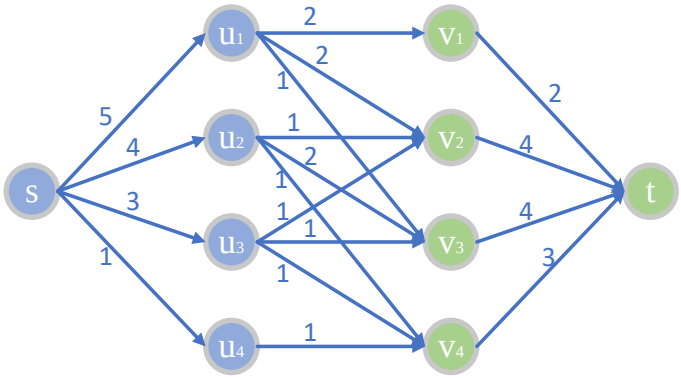


Figure 6. Maximum feasible flow.

The updated traffic matrix is shown in Figure 7. As the updated matrix A still has available capacity, the advance scheduling algorithm will continue to execute until there is no available capacity at the ports of matrix A or the upper and lower bound maximum flow algorithm cannot find the maximum flow, at which point the algorithm terminates.

$$A=\begin{array}{c|cccc|cc} & v_1 & v_2 & v_3 & v_4 & p_i^2 & c_i \\ \hline u_1 & 2 & 3 & 3 & 0 & 8 & 2 \\ u_2 & 2 & 2 & 2 & 2 & 8 & 2 \\ u_3 & 2 & 2 & 3 & 3 & 10 & 0 \\ u_4 & 1 & 1 & 1 & 3 & 6 & 4 \\ \hline q_j^2 & 7 & 8 & 9 & 8 & & \\ e_j & 3 & 2 & 1 & 2 & & \end{array} \qquad B=\begin{array}{c|cccc|cc} & v_1 & v_2 & v_3 & v_4 & p_i^2 & a_i \\ \hline u_1 & 0 & 0 & 0 & 2 & 2 & 0 \\ u_2 & 1 & 0 & 0 & 1 & 2 & 0 \\ u_3 & 1 & 0 & 1 & 0 & 2 & 0 \\ u_4 & 0 & 2 & 1 & 0 & 3 & 1 \\ \hline q_j^2 & 2 & 2 & 2 & 3 & & \\ b_j & 0 & 0 & 0 & 1 & & \end{array}$$

Figure 7. Updated traffic matrices A and B.

The two-class packet scheduling algorithm steps are shown in Algorithm 4, and its time complexity analysis is as follows: In the algorithms for two-class packet scheduling, the time complexity of the advance scheduling algorithm is mainly determined by the construction of the network flow graph and the maximum flow algorithm. Assuming that the time complexity of the maximum flow algorithm is $O(N^a)$, then the time complexity of the entire advance scheduling algorithm is $O(N^2 + K \times N^a)$, where K is the number of iterations. In summary, the algorithm complexity of the two-class packet scheduling algorithms is $O(N^2 + K \times N^a)$.

Algorithm 4 Two-class packet scheduling algorithm.

Input: A, B
Output: M_i
while (**any**($c_i > 0$) | | **any**($e_j > 0$)) **do**
  Called the advance scheduling algorithm
  Update the traffic matrix A: $A = A + B_x$
  Update the traffic matrix B: $B = B - B_x$
  if (No legitimate viable stream was found) **then**
    break
  end if
end while
Schedule A collection in $[0, d_1]$
Schedule B collection in $[d_1 + 1, d_2]$

3.3. Multi-Class Packet Scheduling

Multi-class packet scheduling refers to the scheduling problem where there are three or more different deadlines within the total packet scheduling set S, $S = S^{d_1} \cup S^{d_2} \cup S^{d_3} \cup \ldots \cup S^{d_n}$. Deadline $D = \{d_1, d_2, d_3 \ldots d_n\}$, $d_1 < d_2 < d_3 < \ldots < d_n$. This type of scheduling problem with multiple deadlines has been proven to be an NP-hard problem, making it difficult to find an exact optimal solution [35]. Therefore, the only approach is to approximate the optimal solution.

The multi-class packet scheduling problem is an extension of the two-class packet scheduling problem. It can be reduced by converting the multi-class scheduling problem into a scheduling problem with two deadlines. This is achieved by iteratively invoking two-class packet scheduling algorithms through polling and updating the traffic matrix. This approach resolves the scheduling problem for multiple classes of packets.

Take three-class packet scheduling as an example. First, according to the deadlines, packets in the set are divided into three categories from small to large: high priority S^{d_1}, medium priority S^{d_2}, and low priority S^{d_3}. First, call the advance scheduling algorithm for the S^{d_1} and S^{d_2}. At the same time, if there still exist some free time slots in the S^{d_1}, the advance scheduling algorithm is invoked on the S^{d_1} and S^{d_3} until the input and output ports in S^{d_1} are not free, or the advance scheduling algorithm cannot find the maximum legal flow, then the maximum weight matching algorithm is invoked in the time slot $[0, d_1]$ to decompose the traffic matrix composed of S^{d_1}. Then the set S^{d_2} is regarded as a high-priority set, the set S^{d_3} is regarded as a sub-priority packet set, and the above process is repeated, scheduling the set S^{d_2} packets in the time slot $[d_1 + 1, d_2]$. Finally, the set S^{d_3} is scheduled in the time slot $[d_2 + 1, d_3]$.

The time complexity analysis of multi-class packet scheduling algorithm, as shown in the Algorithm 5, is as follows. The main time complexity is related to the algorithm used to find the maximum flow. The time complexity of the upper and lower bound maximum flow algorithm is $O(E \times maxflow)$, where E is the number of edges in the graph and the $maxflow$ indicates the maximum traffic value in the network. In summary, the overall time complexity of the multi-class packet scheduling algorithm can be approximated as $O(k \times E \times maxflow)$, where k is the number of iterations required to find the maximum traffic in the loop, E is the number of edges in the graph, and the *maxflow* is the maximum traffic value.

Algorithm 5 Multi-class packet scheduling.

Input: $S_1, S_2, S_3 \ldots S_n$, $D = \{d_1, d_2, d_3 \ldots d_n\}$
Output: M_i

```
Calculate the S_1 matrix's emptiness c_i, e_j
Calculate the S_2 matrix's excess a_i, b_j
for l = 1 : n − 1 do
    while c_i > 0 || e_j > 0 do
        for t = l + 1 : n do
            Create network flow G(s, t, U, V, E)
            Called Advance scheduling algorithm (S_l, S_t)
            Output the lift matrix B_x
            Update the traffic matrix, S_l = S_l + B_x
            Update the traffic matrix, S_t = S_t − B_x
            if (No legitimate viable stream was found) then
                break
            end if
        end for
        if (All matrices have been iterated) then
            break
        end if
    end while
    Schedule the S_l collection
end for
Schedule the S_t collection
End
```

4. Simulation and Analysis

We validated the performance of the CSDN-TCS algorithm using software simulation. In this study, simulation experiments were designed from two perspectives: based on the switch size and based on the line load. We compared the QoS performance differences among the traditional heuristic EDF algorithm, the FIPS algorithm, and the CSDN-TCS algorithm under single-class, two-class, and multi-class scenarios (the multi-class experiments are represented by three classes). Specific performance metrics include the success rate, packet loss rate, average delay, and throughput.

4.1. Performance Indicators

In order to study the performance of the algorithm, several indexes such as the success rate, packet loss rate, average delay, and throughput rate are introduced to evaluate the performance of the algorithm.

4.1.1. Success Rate

The success rate is α defined as the number of times the algorithm successfully schedules all the packets in the experiment conducted. In a schedulable set, if only one packet is dropped, the algorithm will also be judged as a failure:

$$\alpha = \frac{success}{total} \tag{7}$$

Success represents the number of successful scheduling and *total* represents the total number of experiments.

4.1.2. Packet Loss Rate

The packet loss rate β is defined as the ratio of the number of discarded packets in a scheduling set to the total number of scheduled packets:

$$\beta = \frac{dropped\ packets}{total\ packets} \tag{8}$$

The *dropped packets* indicates the number of packets discarded in this schedule, and the *total packets* indicates the total number of packets in this scheduling.

4.1.3. Average Latency

The average delay γ is defined as the ratio between the total delay of successfully scheduled packets and the number of successfully scheduled packets:

$$\gamma = \frac{\sum_{i=1}^{deadline} time\ delay}{transmitted\ packets} \tag{9}$$

The $\sum_{i=1}^{deadline} time\ delay$ indicates the sum of the delays of all successfully scheduled packets in the current transmission, and the $transmitted packets$ indicates the total number of packets that are successfully scheduled before the deadline in this transmission.

4.1.4. Throughput Rate

In the experimental simulation, packets that miss the deadline are discarded. We define the throughput μ as the ratio between the number of packets that successfully completed the schedule before the deadline and the total number of packets for this schedule:

$$\mu = \frac{packets\ successfully\ scheduled}{total\ packets} \tag{10}$$

The $packets\ successfully\ scheduled$ indicates the number of packets successfully transmitted to the corresponding output port in this schedule, and the *total packets* indicates the total number of packets transmitted in this schedule.

4.2. Simulation Experiment Based on Switch Size

First, we study the relationship between the switch size and success rate, packet loss rate, average delay and throughput rate under low-load conditions.

The experimental design ideas based on the switching size are as follows (take two types as examples): randomly generate A and B traffic matrices in $N \times N$ size, set deadlines d_1 and d_2 to meet the low-load state of the switching system, and conduct simulation experiments.

From Figure 8, it can be observed that in single-class scheduling, the scheduling success rates of the traditional iSLIP, EDF, MLF algorithms, FIPS algorithm, and CSSDN-TCS algorithm are relatively high when the switch size is small. As the switch size increases, the scheduling success rates gradually decrease. Compared to traditional heuristic algorithms, the success rate performance of the CSDN-TCS algorithm is improved by approximately 10%.

From Figure 9, it can be seen that in single-class scheduling tasks, the average delays of the iSLIP algorithm, EDF, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm are similar. As the switch size increases, the number of groups in the traffic matrix continues to grow. When $N = 32$, the average delay of the CSDN-TCS algorithm is approximately 32 time slots per packet.

In Figure 10, as the switch size increases, the average packet delay shows an upward trend. When the switch size is small, the average delays of the iSLIP algorithm, EDF, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm are similar. However, as the switch size continues to increase $N \geqslant 32$, the differences in the average packet delay between various comparison algorithms gradually become apparent. When $N = 128$, the average delay of traditional iSLIP, EDF, and MLF algorithms exceeds 250 time slots per packet. While the FIPS algorithm shows a slight decrease in average packet delay, the proposed CSDN-TCS algorithm maintains an average delay of no more than 200 time slots per

packet, exhibiting a significant decrease. The average delay performance of the CSDN-TCS algorithm is improved by 25% compared to other comparative algorithms.

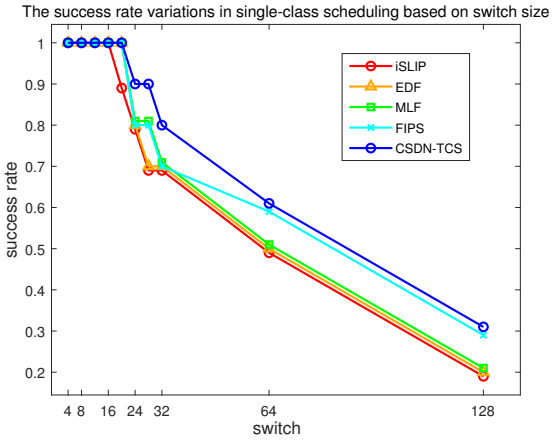


Figure 8. The relationship between the scheduling success rate and switch size in single-class scheduling.

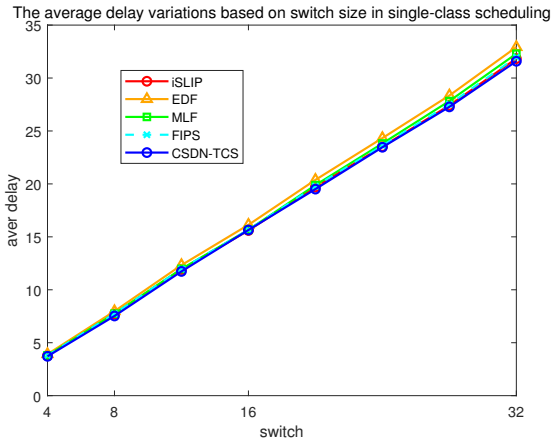


Figure 9. The relationship between average packet delay and switch size in single-class scheduling.

From the graph in Figure 11, it can be observed that the packet loss rate of three-class scheduling decreases with the increase in switch size. From the perspective of the switch, as the switch size increases, the number of input and output ports in the switch also increases. When packets arrive at the switch for scheduling, the probability of conflicts decreases, resulting in a reduction in the packet loss rate. From an algorithmic perspective, for both the FIPS algorithm and CSDN-TCS algorithm, as the switch size increases, larger deadline packets have more opportunities to be scheduled into the idle slots of smaller deadline packets. They are scheduled together with smaller deadline packets, leading to a rapid decrease in the packet loss rate for these two algorithms. When N = 128, the packet loss rate performance of the CSDN-TCS algorithm is improved by approximately 8% compared to iSLIP, EDF, and MLF algorithms.

As can be seen from Figure 12, it is evident that when the switch size is small, the average delay performance of various comparison algorithms is quite similar. The performance of the CSDN-TCS algorithm is slightly better than the others. However, as the switch size increases, the advantage of the CSDN-TCS algorithm in the average packet delay becomes more apparent. When $N = 128$, it is clear that the average delay of the CSDN-TCS algorithm is significantly smaller than the other comparison algorithms. At this point, the average delay performance of the CSDN-TCS algorithm is improved by approximately 15% compared to other comparative algorithms. Moreover, as the switch size grows, the per-

formance advantage of the CSDN-TCS algorithm in the average packet delay becomes more pronounced. Due to the larger number of packets in the traffic matrix of three-class scheduling, leading to higher decision complexity, the average packet delay in three-class scheduling tasks is higher than that in two-class scheduling tasks under the same switch size conditions.

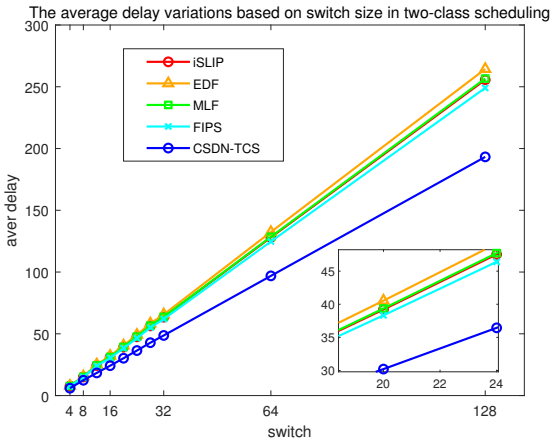


Figure 10. The relationship between average packet delay and switch size in two-class scheduling.

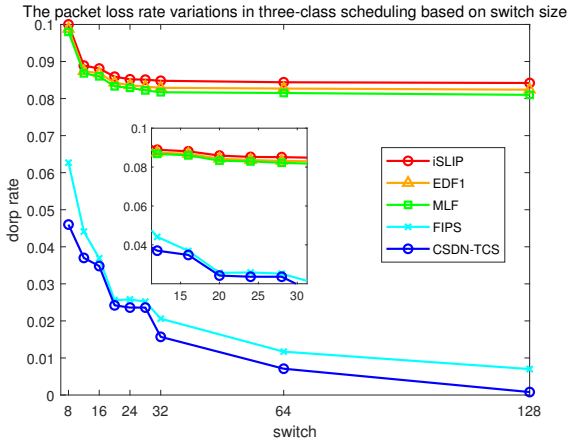


Figure 11. The relationship between packet loss rate and switch size in three-class scheduling.

Figure 13 illustrates the relationship between the throughput and switch size for various comparison algorithms. The results indicate that as the switch size increases, the system's throughput shows an upward trend. When $N \geqslant 32$, the CSDN-TCS algorithm exhibits the highest throughput, followed by the FIPS algorithm, while the traditional iSLIP, EDF, and MLF algorithms have lower throughput. When $N = 128$, the throughput performance of the CSDN-TCS algorithm is improved by approximately 8% compared to traditional iSLIP, EDF, and MLF algorithms.

4.3. Simulation Experiment Based on Line Load

We also study the effect of the line load on the algorithm success rate, packet loss rate, average delay and throughput in 16×16 size. The simulation is carried out for different situations where the line load increases from 0.5 to 1.

The line *load* is defined as the ratio between the total number of packets scheduled for this transmission and the total number of available time slots:

$$load = \frac{total\ packets}{total\ time\ slots} \tag{11}$$

where the *total packets* indicates the total number of packets transmitted in this schedule, and the *total time slots* indicates the total number of time slots that can be utilized before the deadline in this schedule.

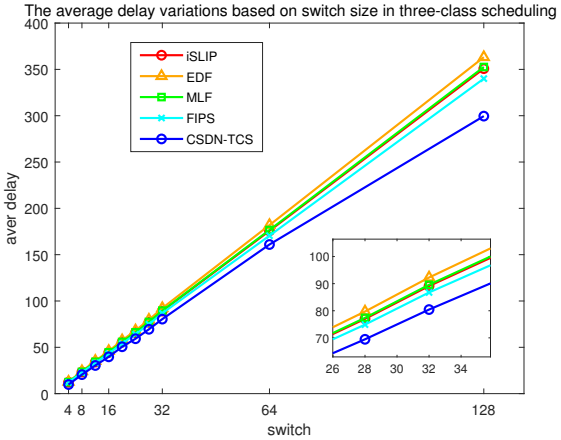


Figure 12. The relationship between the average packet delay and switch size in three-class scheduling.

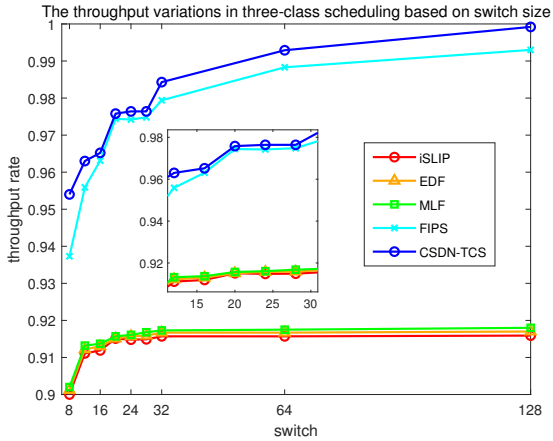


Figure 13. The relationship between the throughput and switch size in three-class scheduling.

Experimental design ideas based on the line load (taking two types as examples) are as follows: Set appropriate deadlines d_1 and d_2. Two 16×16 traffic matrices are randomly generated by $load = 0.5 \ldots 0.9, 1$, respectively, and scheduling experiments are carried out.

The relationship between the algorithmic line load and packet loss rate is depicted in Figure 14. As the line load of the switching system increases, the packet loss rate also increases. When the system's line load is relatively low, the packet loss rates of the iSLIP algorithm, EDF, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm are all at a low level and are relatively close. When $load = 1$, the packet loss rate of the CSDN-TCS algorithm is lower than that of the other compared algorithms, remaining below 6%, while the packet loss rates of the other compared algorithms are all above 10%.

The simulation results for testing the average packet delay of various comparison algorithms are presented in Figure 15. The average packet delay shows a proportional correlation with the line load. When the line load is low, the differences in the average packet delay among the iSLIP algorithm, EDF, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm are relatively small. However, when there is burst traffic in the switching system, leading to a high load situation. When $load = 1$, the CSDN-TCS algorithm exhibits a lower average delay compared to other compared algorithms, with a performance improvement of approximately 5%. This indicates that the algorithm proposed in this paper demonstrates stronger adaptability in high-load conditions.

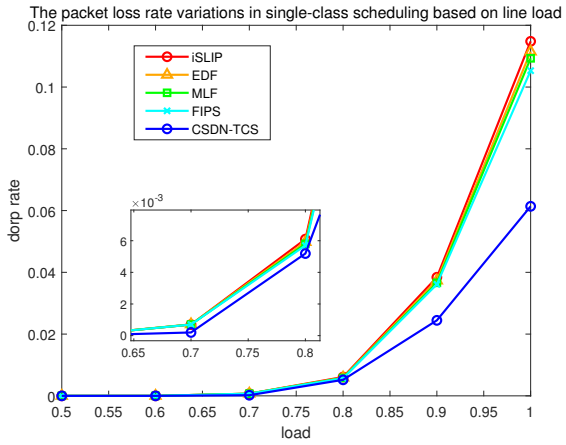


Figure 14. The relationship between the packet loss rate and line load in single-class scheduling.

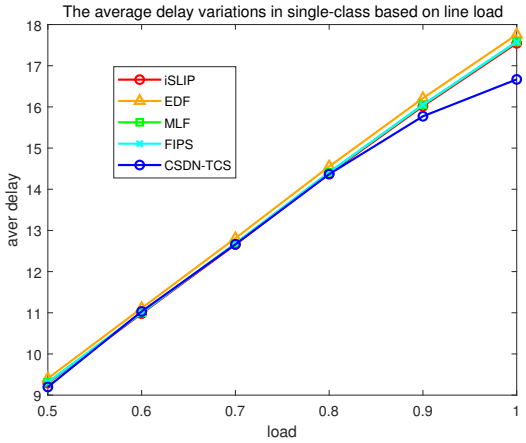


Figure 15. The relationship between average packet delay and line load in single-class scheduling.

From Figure 16, it is evident that in two-class packet scheduling, as the line load increases, the packet loss rate of the switching system also increases. Among them, the iSLIP, EDF, and MLF algorithms exhibit relatively high packet loss rates, with the FIPS algorithm following. The CSDN-TCS algorithm has the lowest packet loss rate, and as the line load increases, the performance advantage of the CSDN-TCS algorithm in the packet loss rate becomes more pronounced. When $load = 0.9$, the packet loss performance of the CSDN-TCS algorithm is improved by approximately 15% compared to the iSLIP, EDF, and MLF algorithms.

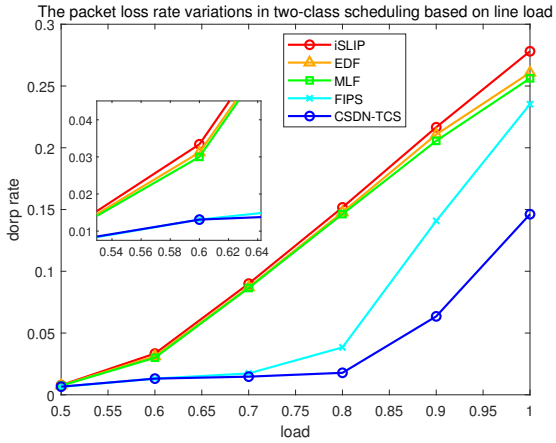


Figure 16. The relationship between the packet loss rate and line load in two-class scheduling.

Figure 17 illustrates the relationship between the system throughput and line load in two-class scheduling tasks. From the figure, it can be observed that when the system load is low, the switching system has a relatively high throughput. However, when $load \geqslant 0.8$, the system's throughput decreases rapidly. Nevertheless, the CSDN-TCS algorithm can still maintain a relatively high throughput compared to the other two algorithms in high-load scenarios. When $load = 1$, the throughput of the CSDN-TCS algorithm is still maintained around 85%, demonstrating the reliability of the proposed algorithm under high-load conditions.

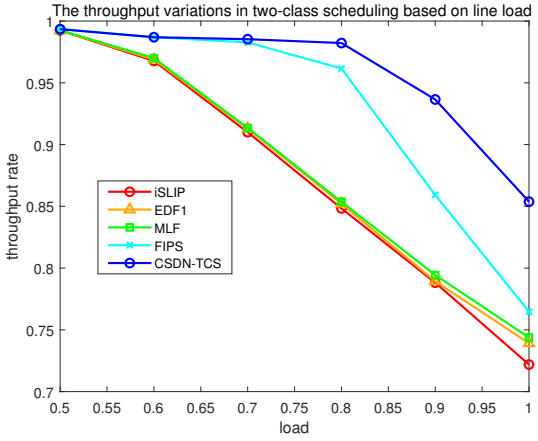


Figure 17. The relationship between the throughput and line load in two-class scheduling.

In Figure 18, the relationship between the packet loss rate and line load in three-class scheduling tasks is depicted. The system's packet loss rate is directly proportional to the line load status. The packet loss rate in multi-class scheduling is higher than in two-class scheduling. In other words, as the number of deadlines in the scheduling set increases, the decision complexity of the scheduling tasks rises, leading to a higher packet loss rate. From the figure, it can be observed that the CSDN-TCS algorithm has the lowest packet loss rate, outperforming the iSLIP algorithm, EDF, MLF algorithm, and FIPS algorithm. When $load = 1$, the packet loss performance of the CSDN-TCS algorithm improves by approximately 10% compared to EDF and MLF algorithms.

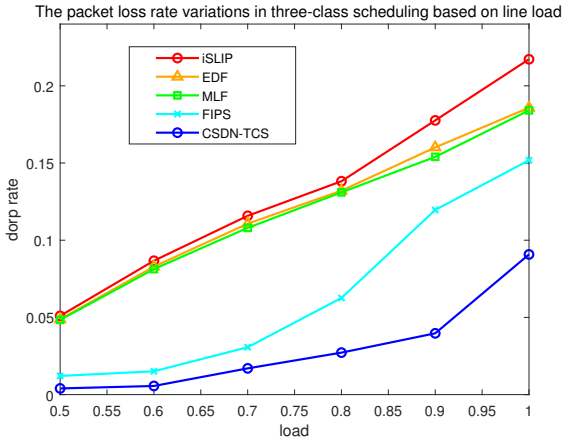


Figure 18. The relationship between the packet loss rate and line load in three-class scheduling.

As can be seen from Figure 19, it is evident that the system throughput is inversely proportional to the line load. As the line load increases, the CSDN-TCS algorithm exhibits the slowest decrease in throughput. Overall, the CSDN-TCS algorithm has the highest throughput, maintaining a level of 90% or above. This is followed by the FIPS algorithm, while the iSLIP, EDF, and MLF algorithms have relatively lower throughput. This indicates

that under the same load conditions, the CSDN-TCS algorithm can schedule more packets with deadlines guaranteed compared to other comparison algorithms.

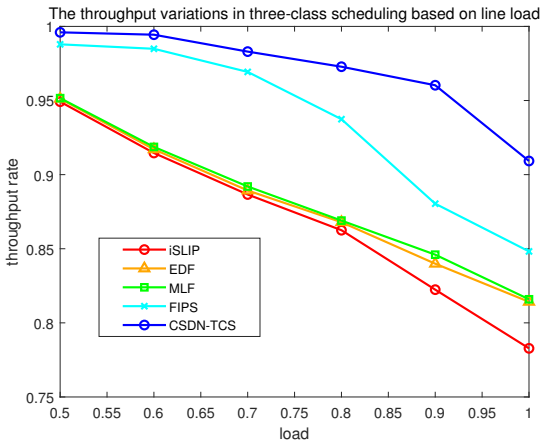


Figure 19. The relationship between the throughput and line load in three-class scheduling.

In summary, we conducted simulation experiments from two aspects: based on system switch size and based on system line load. In single-class, two-class, and multi-class (using three classes as an example) scheduling tasks, we compared the success rate, packet loss rate, average packet delay, throughput, and other QoS performance metrics of the traditional iSLIP algorithm, EDF algorithm, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm. The simulation results indicate that under similar conditions, the CSDN-TCS algorithm can achieve a lower packet loss rate and a smaller average packet delay, ensuring the successful completion of packet scheduling tasks for as many packets as possible before the deadline arrives.

5. Conclusions

Firstly, this paper establishes the model of the input-queuing switch system in TSN, delving into the operational mechanism of input-queuing systems in TSN networks and the scheduling process of packets. Based on this, the paper clarifies the necessity of studying packet scheduling with deadline guarantees. Secondly, the paper proposes the CSDN-TCS algorithm, which builds a network flow model based on the input traffic matrix. It transforms the packet scheduling problem into a network flow problem with upper and lower bounds on the traffic, and solves it using the maximum flow algorithm from graph theory. The algorithm achieves scheduling tasks for single-class, two-class, and multi-class groups with deadline constraints, obtaining results with maximized throughput and minimized average packet delay.

According to the results of simulation experiments, in single-class packet scheduling, the iSLIP algorithm, EDF algorithm, MLF algorithm, FIPS algorithm, and CSDN-TCS algorithm show similar QoS performance when the switch size is small or the line load is low. In two-class and multi-class packet scheduling, especially in scenarios with larger switch sizes or higher line loads, the CSDN-TCS algorithm outperforms other algorithms with higher throughput and smaller average delay. Moreover, under comparable QoS performance, the CSDN-TCS algorithm has lower time complexity than the FIPS algorithm, requiring less scheduling complexity for the traffic matrix and making it more adaptable to general scheduling scenarios.

The research in this paper is still ongoing and requires further refinement of related work. Currently, the research is based on the offline scheduling planning of a single node and has not been extended to global scheduling with multiple nodes, which will be a focus of future work. On the other hand, the research in this paper is more suitable for TSN closed network scenarios, where planning can be done in advance. However, in the face of

dynamically changing real-time networks, the real-time computation of scheduling results may be required, and the proposed algorithm's complexity in this context may be higher.

Author Contributions: Conceptualization, L.Z., G.W. and K.Z.; methodology, L.Z. and G.W.; software, L.Z. and G.W.; validation, L.Z. and G.W.; investigation, L.Z. and G.W.; resources, H.C.; writing—original draft preparation, G.W.; writing—review and editing, L.Z. and G.W.; visualization, L.Z. and G.W.; supervision, K.Z. and H.C.; funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Natural Science Foundation of China under Grant 62102314, and Natural Science Basic Research Program of Shaanxi Province under Grant 2021JQ-708 and 2022JQ-635.

Data Availability Statement: Data is contained within the article.

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

References

1. Romanov, A.M.; Gringoli, F.; Sikora, A. A precise synchronization method for future wireless TSN networks. *IEEE Trans. Ind. Inform.* **2020**, *17*, 3682–3692. [CrossRef]
2. Alaslani, M.; Nawab, F.; Shihada, B. Blockchain in IoT systems: End-to-end delay evaluation. *IEEE Internet Things J.* **2019**, *6*, 8332–8344. [CrossRef]
3. Huang, T.; Wang, S.; Huang, Y.; Zheng, Y.; Liu, J.; Liu, Y. Survey of the deterministic network. *J. Commun.* **2019**, *40*, 160–176.
4. Deng, L.; Xiao, X.; Liu, H.; Li, R.; Xie, G. A low-delay AVB flow scheduling method occupying the guard band in Time-Sensitive Networking. *J. Syst. Archit.* **2022**, *129*, 102586. [CrossRef]
5. Zhang, T.; Feng, J.; Ma, Y.; Qu, S.; Ren, F. Survey on Traffic Scheduling in Time-Sensitive Networking. *Comput. Res. Dev.* **2021**, *59*, 747–764.
6. Xie, G.; Zeng, G.; Liu, L.; Li, R.; Li, K. High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems. *J. Syst. Archit.* **2016**, *70*, 3–14. [CrossRef]
7. Tan, W.; Wu, B.; Wang, S.; Huang, T. Large-scale Deterministic Transmission among IEEE 802.1 Qbv Time-Sensitive Networks. In Proceedings of the ICC 2022—IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 2315–2320.
8. Patti, G.; Bello, L.L.; Leonardi, L. Deadline-aware online scheduling of tsn flows for automotive applications. *IEEE Trans. Ind. Inform.* **2022**, *19*, 5774–5784. [CrossRef]
9. Yun, Q.; Xu, Q.; Zhang, Y.; Chen, Y.; Sun, Y.; Chen, C. Flexible Switching Architecture with Virtual-Queue for Time-Sensitive Networking Switches. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 13–16 October 2021; pp. 1–6.
10. Kulikowski, D. Cross-Layer Design Approach for Dynamic Control over Wi-Fi Network Slices: Bringing QoS Diversity to Wi-Fi Based IoT Networks. Bachelor's Thesis, University of Twente, Enschede, The Netherlands, 2023.
11. Zheng, L.; Qiu, Z.; Sun, S.; Pan, W.; Gao, Y.; Zhang, Z. Design and analysis of a parallel hybrid memory architecture for per-flow buffering in high-speed switches and routers. *J. Commun. Netw.* **2018**, *20*, 578–592. [CrossRef]
12. Waqas Haider Shah, S.; Qaraqe, M. Energy-efficient machine type communication in HetNets under statistical QoS guarantees. *Trans. Emerg. Telecommun. Technol.* **2023**, *34*, e4848. [CrossRef]
13. Kwon, S. Ensuring renewable energy utilization with quality of service guarantee for energy-efficient data center operations. *Appl. Energy* **2020**, *276*, 115424. [CrossRef]
14. Nguyen, H.K.; Tran, X.T. A novel reconfigurable router for QoS guarantees in real-time NoC-based MPSoCs. *J. Syst. Archit.* **2019**, *100*, 101664. [CrossRef]
15. Lee, Y.; Lou, J.; Luo, J.; Shen, X. An efficient packet scheduling algorithm with deadline guarantees for input-queued switches. *IEEE/ACM Trans. Netw.* **2007**, *15*, 212–225. [CrossRef]
16. Zheng, L.; Jiang, J.; Pan, W.; Liu, H. High-performance and range-supported packet classification algorithm for network security systems in SDN. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
17. Gursoy, M.C.; Mitra, U. Energy-efficient packet scheduling under two-sided delay constraints. In Proceedings of the ICC 2023—IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023; pp. 5322–5327.
18. Gu, Y.; Liu, B.; Shen, X. Asymptotically optimal online scheduling with arbitrary hard deadlines in multi-hop communication networks. *IEEE/ACM Trans. Netw.* **2021**, *29*, 1452–1466. [CrossRef]
19. Zhang, B.; Wan, X.; Luo, J.; Shen, X. A nearly optimal packet scheduling algorithm for input queued switches with deadline guarantees. *IEEE Trans. Comput.* **2014**, *64*, 1548–1563. [CrossRef]
20. Martelli, F.; Bonuccelli, M.A. Minimum message waiting time scheduling in distributed systems. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *24*, 1797–1806. [CrossRef]

21. Chi, Y.; Zhang, H.; Liu, Y.; Chen, N.; Zheng, Z.; Zhu, H.; Zhang, P.; Zhan, H. Flow-Based Joint Programming of Time Sensitive Task and Network. *Electronics* **2023**, *12*, 4103. [CrossRef]
22. Torres, J.C. Normality of *k*-Matching Polytopes of Bipartite Graphs. *arXiv* **2023**, arXiv:2306.11910.
23. Zhang, Y.W.; Ma, J.P.; Zheng, H.; Gu, Z. Criticality-Aware EDF Scheduling for Constrained-Deadline Imprecise Mixed-Criticality Systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *43*, 480–491. [CrossRef]
24. Chetto, M.; El Osta, R. Earliest Deadline First Scheduling for Real-Time Computing in Sustainable Sensors. *Sustainability* **2023**, *15*, 3972. [CrossRef]
25. Chen, N.; Kurniawan, C.; Nakahira, Y.; Chen, L.; Low, S.H. Smoothed least-laxity-first algorithm for electric vehicle charging: Online decision and performance analysis with resource augmentation. *IEEE Trans. Smart Grid* **2021**, *13*, 2209–2217. [CrossRef]
26. Nakahira, Y.; Chen, N.; Chen, L.; Low, S.H. Smoothed least-laxity-first algorithm for EV charging. In Proceedings of the Eighth International Conference on Future Energy Systems, Hong Kong, China, 16–19 May 2017; pp. 242–251.
27. Nitnaware, V.; Limaye, S.S. Time Efficient Arbiter in the design of Scheduler embodying Islip algorithm for on chip interconnection. *Int. J. Adv. Sci. Technol. IJAST* **2010**, *21*, 69–82.
28. Wu, J.; Li, B. On the Complexity of Optimal Scheduling Multi-Rate Nested Periodic Trafficin an Input-Queued Switch. *J. Comput. Sci.* **2010**, *33*, 55–62.
29. Majidi, S.; Obermaisser, R. Genetic Algorithm for Scheduling Communication Networks in Time-Triggered Systems-of-Systems. In Proceedings of the 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Wuhan, China, 25–28 October 2021; pp. 121–128.
30. Pahlevan, M.; Obermaisser, R. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, Italy, 4–7 September 2018; Volume 1, pp. 337–344.
31. Wille, E.C.; Hoffmann, J.R. GENIUS–A genetic scheduling algorithm for high-performance switches. *AEU-Int. J. Electron. Commun.* **2015**, *69*, 629–635. [CrossRef]
32. Ran, C.; Su, H.; Sun, Y.; Sun, J. Research on fair scheduling algorithm of high-performance input queuing switch. *Foreign Electron. Meas. Technol.* **2023**, *42*, 114–119.
33. Wan, X.; Shan, F.; Shen, X. An optimal algorithm for time-slot assignment in SS/TDMA satellite systems. In Proceedings of the 2013 22nd International Conference on Computer Communication and Networks (ICCCN), Nassau, Bahamas, 30 July–2 August 2013; pp. 1–6.
34. Shen, X.; Lou, J.; Liang, W.; Luo, J. Deadline guaranteed packet scheduling for overloaded traffic in input-queued switches. *Theor. Comput. Sci.* **2008**, *409*, 477–485. [CrossRef]
35. Raviv, L.O.; Leshem, A. Joint scheduling and resource allocation for packets with deadlines and priorities. *IEEE Commun. Lett.* **2022**, *27*, 248–252. [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.