

Article

Deterministic Scheduling for Asymmetric Flows in Future Wireless Networks

Haie Dou ^{1,2}, Taojie Zhu ^{1,2} , Fei Li ¹, Chen Liu ³ and Lei Wang ^{1,2,*}

¹ School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; douhaie@njupt.edu.cn (H.D.); 2024010126@njupt.edu.cn (T.Z.); lifei@njupt.edu.cn (F.L.)

² Jiangsu Engineering Research Center of Communication and Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

³ School of Electronic and Optical Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; liuch@njupt.edu.cn

* Correspondence: wanglei@njupt.edu.cn

Abstract

In the era of Industry 5.0, future wireless networks are increasingly shifting from traditional symmetric architectures toward heterogeneous and asymmetric paradigms, driven by the demand for diversified and dynamic services. This architectural evolution gives rise to complex and asymmetric flows, such as the coexistence of periodic and burst flows with varying latency, jitter, and deadline constraints, posing new challenges for deterministic transmission. Traditional time-sensitive networking (TSN) is well-suited for periodic flows but lacks the flexibility to effectively handle dynamic, asymmetric traffic. To address this limitation, we propose a two-stage asymmetric flow scheduling framework with dynamic deadline control, termed A-TSN. In the first stage, we design a Deep Q-Network-based Dynamic Injection Time Slot algorithm (DQN-DITS) to optimize slot allocation for periodic flows under varying network loads. In the second stage, we introduce the Dynamic Deadline Online (DDO) scheduling algorithm, which enables real-time scheduling for asymmetric flows while satisfying flow deadlines and capacity constraints. Simulation results demonstrate that our approach significantly reduces end-to-end latency, improves scheduling efficiency, and enhances adaptability to high-volume asymmetric traffic, offering a scalable solution for future deterministic wireless networks.

Keywords: asymmetric flows; quality of service; Deep Q-Network; injection slot offset; time-sensitive networking



Academic Editor: Chengxi Zhang

Received: 28 May 2025

Revised: 16 July 2025

Accepted: 18 July 2025

Published: 6 August 2025

Citation: Dou, H.; Zhu, T.; Li, F.; Liu, C.; Wang, L. Deterministic Scheduling for Asymmetric Flows in Future Wireless Networks. *Symmetry* **2025**, *17*, 1246. <https://doi.org/10.3390/sym17081246>

Copyright: © 2025 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

The rapid evolution of industrial automation and intelligent transportation systems has led to the widespread deployment of heterogeneous networked control architectures featuring asymmetric design principles. In such asymmetric architectures, computational resources, sensing modules, and actuators are distributed across multiple hierarchical layers with differing capabilities and responsibilities [1]. This architectural asymmetry results in non-uniform traffic patterns comprising both periodic control flows and burst flows triggered by stochastic events [2,3].

Non-asymmetric traffic, particularly burst flows, plays a pivotal role in ensuring system safety and responsiveness. For instance, emergency braking signals in vehicular networks or equipment fault alarms in industrial factories require ultra-low latency and

deterministic transmission guarantees to prevent catastrophic consequences [4]. Meanwhile, periodic flows, such as sensor status updates and routine control commands, demand bounded delay with high reliability to maintain system stability. Therefore, the efficient scheduling of these asymmetric flows is crucial for the real-time performance and safety of physical systems [5,6].

Time-sensitive networking (TSN) has emerged as a key enabling technology to address the deterministic communication requirements of such heterogeneous systems [7]. By integrating standards such as the Time-Aware Shaper (TAS) and Cyclic Queuing and Forwarding (CQF), TSN provides bounded end-to-end delay and minimal jitter for periodic traffic [8]. TAS reserves time slots for high-priority flows to ensure ultra-low latency, while CQF leverages cycle-based queuing to guarantee bounded delay for large-volume periodic flows. However, these mechanisms are inherently limited in handling burst flows with unpredictable arrival times and strict real-time deadlines, as they rely on pre-configured slot allocation without dynamic adaptability [9,10].

Recent research has proposed heuristic-based scheduling algorithms to optimize periodic flow scheduling [11]. More recently, reinforcement learning (RL)-based approaches have been introduced to enhance adaptability in TSN scheduling [12–14]. Nevertheless, these methods predominantly focus on periodic flows or gate control scheduling, lacking a unified framework capable of jointly optimizing periodic and burst flows in mixed-traffic environments [15]. Moreover, they rarely integrate offline injection slot allocation with online deadline-aware adjustments, limiting their effectiveness in real-time asymmetric flow scheduling scenarios.

To address these limitations, this paper proposes A-TSN, a novel two-stage asymmetric flow scheduling framework that integrates offline Deep Q-Network-based Dynamic Injection Time Slot allocation (DQN-DITS) with online Dynamic Deadline Optimization (DDO). The main contributions of this work are as follows:

- We design a Deep Q-Network-based injection slot allocation algorithm for periodic flows, formulating slot assignment as a Markov Decision Process to maximize scheduling success rates while minimizing latency.
- We propose a Dynamic Deadline Online algorithm for burst flows, introducing a deadline adjustment mechanism that dynamically prioritizes flows based on urgency and queuing delay, ensuring real-time transmission guarantees.
- We develop and evaluate the unified A-TSN framework, which integrates DQN-DITS and DDO to achieve deterministic scheduling for mixed periodic and burst flows. Extensive experiments demonstrate up to 160% improvement in scheduling success rate and 40-slot reduction in average flow latency compared to state-of-the-art baselines.

The remainder of this paper is organized as follows: Section 2 reviews related work on TSN scheduling. Section 3 presents the system model. Section 4 details the proposed A-TSN framework and its algorithmic components. Section 5 provides experimental results and analysis. Section 6 concludes the paper.

2. Related Work

This section reviews existing research on the TSN scheduling of periodic flows as well as scheduling approaches for burst flows. We also summarize their limitations and discuss the motivation for this paper.

TSN standards provide various scheduling mechanisms to guarantee deterministic communication for periodic flows. The TAS defines transmission gates with precise opening times to ensure ultra-low latency for high-priority flows [16]. The CQF mechanism utilizes cycle-based queuing to provide bounded delay for large-volume periodic flows. While

the CQF mechanism reduces computational complexity and ensures deterministic delays, it suffers from lower control accuracy and higher packet delays compared to TAS [17]. In summary, while TAS offers precise scheduling for high-priority, time-sensitive flows, it grapples with high computational complexity. Conversely, CQF simplifies scheduling but at the cost of increased packet delays and reduced control accuracy.

To support the mixed transmission of TSN periodic and high-bandwidth flows, hybrid TSN designs combine TAS and CQF mechanisms [18]. As shown in Figure 1, the switch has multiple input ports and queues before the output. TAS queues have the highest priority of 7, CQF queues have priorities of 6 and 5, and best-effort traffic uses the lowest-priority queue. Queue egress is controlled by the Gate Control List (GCL), which schedules flows by priority when multiple gates are open. This hybrid approach balances TAS's precise scheduling for time-sensitive flows with CQF's simplified deterministic delays, ensuring efficient and reliable network transmission.

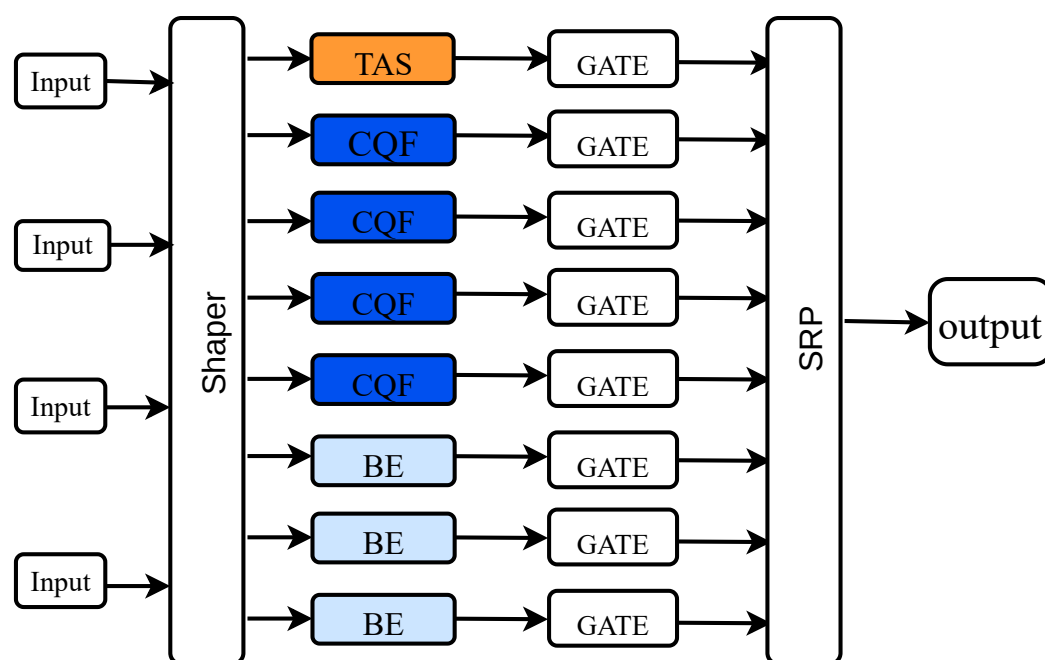


Figure 1. Hybrid TSN system architecture with multiple input nodes (representing different sources) integrating TAS and CQF for periodic flow scheduling.

To improve scheduling efficiency, various flow injection allocation mechanisms have been proposed. Some prioritize flows by urgency through time slot injection planning, enhancing the schedulability of time-sensitive traffic [19]. Dynamic service injection algorithms with adaptive bandwidth allocation adjust reserved slots in real time based on flow arrival rates, optimizing scheduling granularity [20]. Heuristic approaches like FITS pre-allocate slots using fixed priority rules, while HSTCS combines TAS and CQF with simulated annealing to optimize injection slots for medium-priority flows [19]. Tabu search-based methods formulate slot allocation as an optimization problem to avoid local optima and improve schedulability [21]. Although lightweight and easy to implement, these heuristics lack adaptability to dynamic traffic, often resulting in suboptimal resource use under varying conditions.

RL has recently gained attention for TSN scheduling due to its ability to learn optimal policies via environment interaction. Two-stage RL frameworks have been developed to jointly coordinate resource orchestration and transmission scheduling, improving network efficiency and balancing resource allocation [22,23]. Hybrid approaches integrating deep

reinforcement learning with CQF mechanisms have demonstrated effectiveness in both offline and online scheduling scenarios [24]. However, most RL-based TSN scheduling algorithms focus solely on periodic flows or gate control optimization without addressing the integration of offline slot allocation with online scheduling adjustments. Additionally, they do not provide unified scheduling frameworks capable of handling mixed periodic and burst flows in real-time industrial or vehicular environments.

Efforts have also targeted burst traffic mitigation and prioritization. Recent works have explored resource reservation or preemption-based methods to prioritize burst flows [25], but these approaches often lead to bandwidth under-utilization or disrupt scheduled periodic traffic. TDMA-based flow reservation strategies alleviate burst flow impacts but often suffer from bandwidth inefficiency [26]. More flexible scheduling methods adopting earliest-deadline-first and burst-prioritized schemes better handle mixed periodic and burst traffic, enhancing overall system responsiveness [27,28].

In summary, while existing heuristic-based and RL-based TSN scheduling algorithms have achieved notable progress in periodic flow scheduling, they lack unified solutions for mixed traffic environments comprising both periodic and burst flows. Furthermore, current burst flow scheduling approaches fail to provide the dynamic deadline-aware adjustments necessary for strict real-time guarantees.

To address these limitations, this paper proposes A-TSN, a two-stage asymmetric flow scheduling framework that combines offline DQN-based injection slot allocation for periodic flows with online dynamic deadline optimization for burst flows. This unified design ensures deterministic scheduling with improved adaptability and real-time performance under heterogeneous traffic conditions.

3. System Model

Since the transmission period of periodic flows is fixed, the injection slots can be pre-allocated through the injection slot allocation mechanism using the existing TAS and CQF mechanisms to ensure deterministic and reliable end-to-end transmission. However, it becomes challenging to pre-allocate injection slots like periodic flows because the generation slots of burst flows are random. But burst flows are often generated by emergency events (such as tool damage and vehicle emergency braking) and have ultra-low latency and deterministic transmission requirements. If they cannot be processed within a limited time, the consequences will be disastrous. The burst flow scheduling method based on resource reservation will waste bandwidth resources and even hinder the transmission of periodic flows. Therefore, it is an urgent issue to ensure the low-latency end-to-end transmission of periodic flows while meeting the needs of burst flows. To solve these problems, we analyze periodic flow and asymmetric flow separately. Table 1 lists all abbreviations used in this paper.

Table 1. List of abbreviations.

Abbreviation	Definition
TSN	Time-Sensitive Networking
TAS	Time-Aware Shaper
CQF	Cyclic Queuing and Forwarding
GCL	Gate Control List
SRP	Scheduling Resource Pool
HP	High Priority (Flow)
MP	Medium Priority (Flow)
AP	Asymmetric Flow / Burst Flow
A-TSN	Asymmetric Time-Sensitive Networking Framework
DQN	Deep Q-Network

Table 1. Cont.

Abbreviation	Definition
DITS	Dynamic Injection Time Slot
DQN-DITS	Deep Q-Network-based Dynamic Injection Time Slot Algorithm
DDO	Dynamic Deadline Online Algorithm
MDP	Markov Decision Process
RL	Reinforcement Learning
PPO	Proximal Policy Optimization
DDPG	Deep Deterministic Policy Gradient
HSTCS	Hybrid Slot Time Control Scheduling
FITS	Fixed Injection Time Slot Scheduling
D-TSN	Deterministic TSN Scheduling
MTU	Maximum Transmission Unit

3.1. Priority-Aware Granular Scheduling Model for Periodic Flows

According to the scheduling requirements of periodic flows, we define a periodic flow as a tuple that encapsulates the characteristics required for its scheduling. The representation of the tuple is as follows

$$f_i^p = (T_i^p, S_i^p, D_i^p, L_i^p, P_i^p), \quad \forall f_i^p \in F^p, \quad (1)$$

where T_i^p is the flow period, S_i^p is the length of the flow packet, D_i^p is the flow deadline, L_i^p is the flow injection slot, and P_i^p is the flow priority.

Based on the deadline property of periodic flow, the periodic flows are divided into the high-priority flow, called *HP*, and the medium-priority flow, called *MP*, as shown in Table 2. We use TAS to transport the *HP* flows and CQF to transport the *MP* flows.

Table 2. Periodic flow division.

Type	Feature	Adoption Mechanism	Slot Offset
<i>HP</i> Flow	Ultra-low latency and small quantity	TAS	N
<i>MP</i> Flow	Low latency and large quantity	CQF	Y

We introduce the concept of the hyper period T_s , which defines the minimum time range required to successfully schedule all periodic flows. When each periodic flow can be assigned an injection time slot that meets all constraints within T_s , the scheduling of periodic flows is deemed successful, denoted as

$$T_s = LCM(T_1^p, T_2^p, \dots, T_n^p), \quad (2)$$

where LCM means finding the least common multiple.

Simultaneously, the periodic flow utilizes the GCL method to determine the transmission of the flow. In this regard, we introduce the concept of a unit slot T_u to delineate the scheduling granularity, specifically, the GCL update interval. The unit time slot is selected from the factor set of the greatest common multiple of all periodic flows, and the unit time slot is selected to ensure that all the currently cached data packets can be forwarded, which is described as

$$T_g = GCD(T_1^p, T_2^p, \dots, T_n^p), \quad (3)$$

$$\begin{cases} L = \text{factor}(T_g) \\ T_u \geq \frac{MC}{B} + \delta_{\max}, T_u \in L, \end{cases} \quad (4)$$

where GCD stands for the Greatest Common Factor, L is the factor set of T_g , MC is the maximum queue buffer, B is the link transmission rate, and $\delta_{\max}$ is the maximum delay due to time synchronization and other situations.

At the same time, we define the bandwidth occupancy table Q_b and the cache queue occupancy table Q_w of the sending flow based on the hyper cycle T_s and the unit time slot T_u , represented as

$$Q_b(q) = \sum_{i=0}^n \frac{S_i^p}{BT_u} + \sum_{j=0}^m \frac{S_j^p}{BT_u}, \quad (5)$$

$$Q_w(q) = \sum_{j=0}^m \frac{S_j^p}{MC}, \quad (6)$$

where the subscripts i and j represent the HP flow and MP flow, respectively, and q represents the slot.

According to the above definition, we give the optimization problem model for periodic flow, expressed as

$$\begin{aligned} & \max \sum_{i=0}^n F_{suc}(f_i^p) \\ & \text{s.t.} \quad \begin{cases} 0 < q < D_j^p \\ \sum_{i=1}^n S_i^p \times \Phi_T(f_i^p) \leq T_u \times B, \\ 0 \leq Q_b(q) \leq 1 \\ 0 \leq Q_w(q) \leq 1 \end{cases} \end{aligned} \quad (7)$$

where F_{suc} indicates whether the flow is scheduled successfully and Φ_T signifies whether the flow is being transmitted.

3.2. Unified Deadline-Constrained Scheduling Model for Asymmetric Flows

At the beginning of each time slot, the burst flow and the pre-processed periodic flow will enter the SRP. At this moment, the periodic flow can be considered as the burst flow generated during the current time slot. The attributes are consequently redefined as follows

$$f_k^h = \{e_k^h, s_k^h, d_k^h, t_k^h\}, \quad \forall f_k^h \in F^H, \quad (8)$$

where e_k^h symbolizes the slot each flow enters into the SRP. For the periodic flow, it denotes the optimal injection slot determined for each flow in each period. For the burst flow, e_k^h expresses the slot at which it occurs. t_k^h represents the sending time in the SRP for the flows.

To determine the mixed flows transmitted in each time slot, three constraints must be satisfied simultaneously: the flow delay must not exceed its deadline, the total size of transmitted flows must not exceed the link capacity C , and the minimum deadline of transmitted flows F_y^H must be earlier than the maximum deadline of non-transmitted

flows F_n^H . Accordingly, the optimization model for selecting the transmitted flow set in the current time slot is formulated as

$$\begin{aligned} \max \quad & \sum_{k=0}^K F_{suc}(f_k^h) \\ \text{s.t.} \quad & \begin{cases} t_k^h \leq d_k^h \\ \sum_{k=0}^K s_k^h < C \\ \min(F_y^H) > \max(F_n^H). \end{cases} \end{aligned} \quad (9)$$

4. Two-Stage Asymmetric Flow Scheduling Framework

TSN aims to provide deterministic low-latency communication for industrial automation and vehicular networks. However, existing standards such as TAS and CQF primarily address periodic flows and are insufficient for handling burst flows with unpredictable arrival times and ultra-low latency requirements. To bridge this gap, we propose a two-stage asymmetric flow scheduling framework, termed A-TSN, that integrates deep reinforcement learning with real-time deadline-aware adjustments to ensure deterministic guarantees for both periodic and burst flows.

The first stage employs a DQN-DITS algorithm to statically allocate optimal injection slots for periodic flows offline, ensuring a baseline scheduling plan. The second stage uses a DDO algorithm to dynamically adjust transmission decisions for mixed flows in real time, guaranteeing end-to-end delay constraints. The framework architecture is shown in Figure 2.

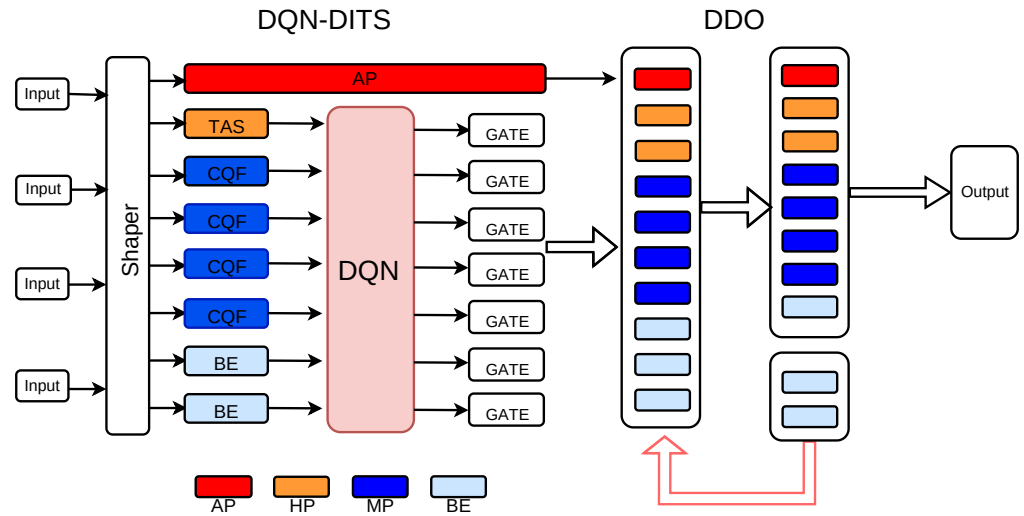


Figure 2. A-TSN: Two-stage asymmetric flow scheduling framework with multiple input nodes (representing different sources) integrating the DQN-DITS and DDO algorithms.

Flows are first filtered and classified into queues according to their properties. Periodic flows are assigned to multiple priority queues managed by TAS and CQF, with queue egress controlled by the GCL. Burst flows bypass the GCL and use no-wait transmission. The DQN-DITS algorithm pre-allocates injection slots for periodic flows and configures the GCL. At each time slot, burst flows and pre-scheduled periodic flows are sent to the SRP for final transmission decision-making via the DDO algorithm.

4.1. Deep Q-Network Framework for Scheduling Decision

DQN (Deep Q-Network) is a reinforcement learning algorithm that combines Q-learning with deep neural networks to approximate action–value functions. It employs a neural network that takes the system state as input and outputs Q-values for each possible action. To ensure training stability, DQN uses two networks: an online network for action selection and a target network for generating stable Q-value targets, which is periodically updated from the online network to mitigate learning oscillations and divergence.

We choose DQN over other reinforcement learning algorithms such as Proximal Policy Optimization (PPO) or Deep Deterministic Policy Gradient (DDPG) because our scheduling problem involves a discrete action space selecting injection time slots. DQN is specifically suited for discrete actions and offers stable, efficient convergence through its experience replay and target network mechanisms. In contrast, DDPG is designed for continuous action spaces, while PPO, despite supporting discrete actions, requires more complex implementations and extensive hyperparameter tuning. Therefore, DQN strikes an effective balance between decision accuracy and computational efficiency, making it the practical choice for real-time injection slot scheduling in time-sensitive networking environments.

Based on the above technologies, we integrate the DQN mechanism into the periodic flow injection slot offset allocation algorithm to achieve the low-latency and reliable transmission of periodic flows, as illustrated in Figure 3. The core components of this design are detailed below:

Environment: The state space is defined as a combination of injection time slots and bandwidth occupancy, with its dimension determined by the hyper period T_s . Formally, it is expressed as

$$S = \{Q_b(0), Q_b(1) \dots, Q_b(a) \dots\}, a \in T_s, \quad (10)$$

where $Q_b(a)$ is the bandwidth occupancy of the time slot a .

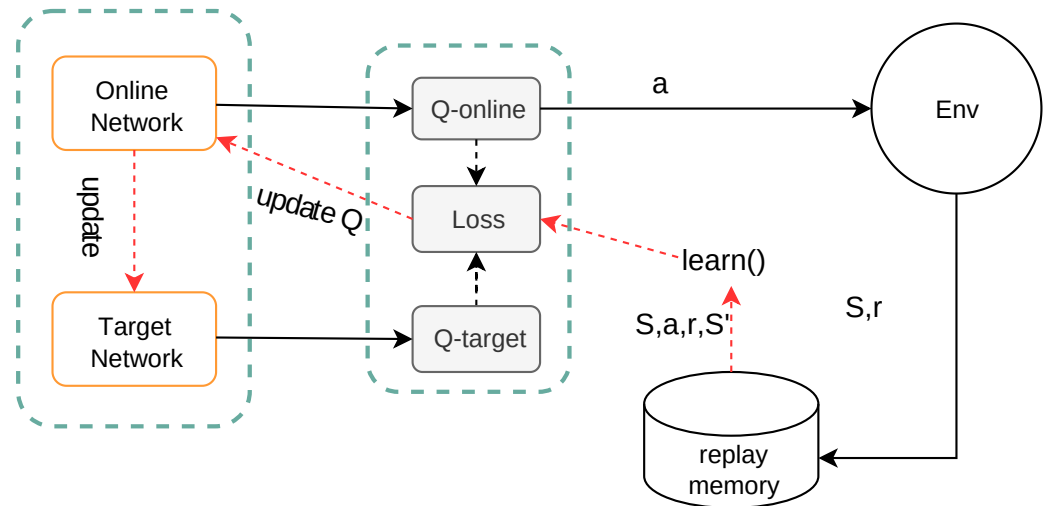


Figure 3. Deep Q-Network architecture for periodic flow injection slot allocation.

Directly using the full state space S as input is both resource-intensive and unnecessary for per-period scheduling. To address this, we extract a schedulable state subspace S_i for each flow in its current period. The length of S_i spans from the start time of the current period to the flow's deadline, focusing only on relevant time slots that are critical for scheduling decisions, expressed as

$$S_i = \{Q_b(\text{start}), Q_b(\text{start} + 1), \dots, Q_b(\text{end})\}, \quad i \in [\text{start}, \text{end}], \quad (11)$$

where start denotes the starting time slot of the current period (e.g., start = t for a flow with period T_j^p , where t is the initial injection time of the period); end is the deadline time slot of the current period (i.e., end = start + D_i^p , with D_i^p representing the deadline of the flow in this period); and $Q_b(i)$ is the bandwidth occupancy ratio of time slot i (normalized to $[0, 1]$).

This subspace S_i effectively narrows the input scope to the time window critical for scheduling decisions, reducing computational overhead while preserving the essential information for optimizing injection slot allocation.

Action: After the agent interacts with the state space selected in each period of each flow S_i , the action a with the optimal Q value is obtained as input. In the action selection process, we use a random greedy algorithm and introduce a probability epsilon ϵ . Under a certain probability, a random action is selected instead of selecting the action based on the current optimal Q value. The action mapping strategy is as follows:

$$a = \begin{cases} \arg \max Q(S_p, a_{f_t}), x < \epsilon \\ a_{f_t}, x \geq \epsilon, \end{cases} \quad (12)$$

where x is a randomly generated constant between $[0, 1]$ and a_{f_t} represents the time interval allocated to flow to inject, described as

$$0 \leq a_{f_t} \leq \frac{T_i^p}{T} - 1, a_{f_t} \in \mathbb{Z}. \quad (13)$$

The Q-value $Q(S_p, a_{f_t})$ represents the long-term revenue for mapping from S_p to a_{f_t} , which is calculated by the action-value function $Q^\pi(s, a)$, described by

$$Q^\pi(s, a) = E \left[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{f_{t+1}}) \mid s_t = s, a_{f_t} = a, \pi \right]. \quad (14)$$

Reward: For the selected action, a reward r is obtained through interaction with the environment. The reward r consists of two parts. r_1 is the distance of the action relative to the deadline, and r_2 is the remaining space of the bandwidth occupancy associated with the current injection time slot. Therefore, the reward can be described as

$$\begin{cases} r = \mu_1 \times r_1 + \mu_2 \times r_2 \\ r_1 = \frac{(D_i^p - a)}{D_i^p}, \\ r_2 = 1 - Q_b(a) \end{cases} \quad (15)$$

where μ_1 and μ_2 are the weights of the two rewards, respectively.

In the Markov Decision Process (MDP) formulation of DQN-DITS, the state S_i represents the schedulable bandwidth occupancy window for each flow. The action space A is the set of discrete injection slot offsets within the flow's period. The reward R combines slot deadline proximity and residual bandwidth capacity, guiding the agent to allocate slots that maximize schedulability while minimizing latency and congestion.

4.2. DQN-DITS Algorithm

For fixed periodic flows, the injection slot offset algorithm can allocate injection slots to all flows before the overall scheduling begins. Our goal is to solve the optimal injection slot allocation problem for periodic flows. Considering the constraints of periodic flows, we define the hybrid period T_s as the total scheduling length and the time slot as T_u . Based on each periodic flow's period and deadline, we normalize the number of offset slots by dividing the deadline by the time slot. Then, all periodic flows are sorted in

descending order according to their bandwidth requirements. Within each period, each flow selects the optimal injection time slot q^* based on its deadline, bandwidth occupancy, and buffer occupancy.

Leveraging a Deep Q-Network (DQN), we propose a learning-based algorithm named DQN-DITS, which integrates the TSN periodic flow injection time slot mechanism with reinforcement learning. Through interaction iterations between the agent and environment, the algorithm effectively learns the mapping between injection time slots and bandwidth occupancy, enabling optimal slot allocation for each periodic flow.

The training process is outlined in Algorithm 1. The algorithm takes as input all periodic flows, initializes an empty replay buffer D , and sets up two neural networks with parameters θ and θ' . At the start of each training episode, the environment state S is reset. For each flow within each period, a state slice S_i is extracted from S according to the deadline and time slot processing rules. The state S_i is fed into the training network parameterized by θ , which outputs an action a following an ϵ -greedy policy based on the optimal Q-value. After executing a , the agent observes the reward r , next state S' , and a terminal signal *done*, indicating whether the current episode ends. The transition $(S, a, r, S', done)$ is stored in the replay buffer D , and the exploration rate ϵ is updated accordingly. Mini-batches sampled from D are used to train the network by minimizing the mean squared error between the target Q-value computed using the target network θ' and the predicted Q-value from the training network. The target network parameters θ' are updated every M steps.

Algorithm 1 Training of the DQN-DITS Algorithm

Require: Periodic flows F^C , empty replay buffer D , initial network parameters θ, θ' , batch size D_b

Ensure: Optimal policy Q

- 1: **for** each episode in episodes **do**
- 2: Reset environment state S
- 3: **for** each flow f^C in F^C **do**
- 4: **for** each period in periods **do**
- 5: Extract input state S_i from S based on deadline and time slot pre-processing
- 6: Obtain action a from policy Q using ϵ -greedy strategy
- 7: Execute action a , receive reward r , next state S' , and done flag
- 8: Store transition $(S, a, r, S', done)$ into replay buffer D
- 9: Update exploration rate ϵ
- 10: **if** *done* **then**
- 11: **break**
- 12: **end if**
- 13: **end for**
- 14: **end for**
- 15: Sample batch of size D_b from replay buffer D
- 16: Set $y_j = \begin{cases} r_j & \text{if } done \\ r_j + \gamma \max_{a'} \hat{Q}(S'_j, a_j; \theta^-) & \text{otherwise} \end{cases}$
- 17: Perform gradient descent on loss $(y_j - Q(S_j, a_j; \theta))^2$ to update θ
- 18: Every M steps, update target network: $\theta' \leftarrow \theta$
- 19: **end for**

After training, the DQN network is integrated into the periodic flow scheduling process, as detailed in Algorithm 2. The algorithm accepts the periodic flow set F^P , the trained network Q , the injection slot set v , and hyperparameters. Initially, the hybrid period T_s and unit slot T_u are computed based on F^P , followed by initializing the bandwidth occupancy Q_b and buffer occupancy Q_w . For each flow in each period, the current bandwidth occupancy state s is extracted and input to the network Q to obtain the injection slot action a .

If the injection satisfies the bandwidth and buffer constraints, a is added to the injection slot set v . This procedure repeats until all flows are processed or constraints are violated. Finally, the algorithm outputs v , representing the injection schedule of all feasible flows.

Algorithm 2 DQN-DITS Scheduling Algorithm

Require: Periodic flows F^P , window size ws , buffer B , network Q , injection slot set v

Ensure: Injection slots v

```

1: Calculate super-period  $T_s$  and unit slot  $T_u$ 
2: Initialize bandwidth  $Q_b$  and window  $Q_w$  according to  $T_s$  and  $T_u$ 
3: Sort flows in  $F^P$  (e.g., by packet length descending)
4: for each flow  $f_i^p$  in  $F^P$  do
5:   for each period  $p$  in  $f_i^p$ .periods do
6:     Extract current state  $s$ 
7:     Select action  $a \leftarrow Q(s)$ 
8:     Update  $Q_b$ ,  $Q_w$  according to action  $a$ 
9:     if slot allocation invalid (!check()) then
10:      break
11:   end if
12:   Store action  $a$  into injection slot set  $v$ 
13: end for
14: end for
15: return  $v$ 

```

4.3. Dynamic Deadline Online Algorithm

While the DQN-DITS algorithm efficiently addresses scheduling for periodic flows, the challenge of mixed flow scheduling in the SRP is handled by the DDO algorithm. Based on the injection time slots determined for periodic flows, DDO calculates which flows can be transmitted in the current time slot.

The core of the DDO algorithm lies in its explicit dynamic deadline adjustment mechanism, which updates the deadlines of unscheduled flows based on their queuing time and original constraints to ensure real-time guarantees. Specifically, for each flow $f_k^h \in F_n^H$ (buffer set), its adjusted deadline $D_k^{h,new}$ is calculated as

$$D_k^{h,new} = \max\left(\alpha \cdot D_k^h - \beta \cdot (t_{\text{current}} - t_k^{\text{arrival}}), D_{\min}\right), \quad (16)$$

where D_k^h is the original deadline of flow f_k^h (initial value set by application requirements); t_{current} is the current time slot start time; t_k^{arrival} is the arrival time of flow f_k^h in the SRP; α ($0 < \alpha < 1$) means the deadline preservation coefficient (weights the original deadline priority); β ($\beta > 0$) is the waiting time penalty coefficient (weights the impact of queuing delay); and $D_{\min}$ is the minimum allowable deadline to prevent negative or impractically small deadlines.

This adjustment ensures that flows delayed in the buffer receive *stricter deadlines* (smaller $D_k^{h,new}$) as their waiting time increases, forcing them to be prioritized in subsequent scheduling cycles to avoid excessive latency.

Algorithm 3 takes as input the stored flows F_k^H in the SRP and outputs the set of flows F_y^H determined for transmission. Within the hybrid period T_s , at the beginning of each time slot, F_k^H is sorted by deadline. Flows f_k^h are iteratively checked against constraints; those meeting requirements are added to the transmission set F_y^H , while those that violate constraints are placed into the buffer set F_n^H , causing the selection loop to break. The waiting times and deadlines of flows in F_n^H are updated accordingly. At the next time slot, F_n^H is merged with newly arrived flows to reform F_k^H , and the scheduling cycle continues.

Algorithm 3 DDO Algorithm for Flow Scheduling**Require:** Flows F^H in SRP, link capacity C , current time $t_{current}$ **Ensure:** Scheduled flows set F_y^H

```

1: while  $T \leq T_s$  do
2:   Sort flows  $F^H$  by deadline ascending
3:   for each flow  $f_k^h \in F^H$  do
4:     if bandwidth occupancy check  $Q_b \leq 1$  passes then
5:       Update bandwidth occupancy
6:       Add  $f_k^h$  to  $F_y^H$ 
7:     else
8:       Add  $f_k^h$  to remaining flows  $F_n^H$ 
9:     break
10:  end if
11: end for
12: Update remaining flows' waiting time and deadlines
13: Return  $F_y^H$ 
14: Repeat loop
15: end while

```

While the DQN-DITS algorithm efficiently addresses scheduling for periodic flows, the challenge of mixed flow scheduling in the SRP is handled by the DDO algorithm. Based on the injection time slots determined for periodic flows, DDO dynamically adjusts flow deadlines and selects transmissible flows in real time.

5. Experimental Evaluation

5.1. Experimental Parameter Settings

We set the maximum frame length MTU to 1500 B, the maximum queue length $ws = 6MTU$, the link rate $B = 1000 \text{ Mbits}^{-1}$, and $\delta_{\max} = 1 \mu\text{s}$. For *HP* flows, the maximum adoption period and packet length are randomly selected from the sets $\{0.6, 0.8, 1, 1.2, 1.6\}$ ms and $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ kB, respectively. The minimum period for *HP* flows is set to 0.1 ms, and the deadline for *HP* flows is set to one twentieth of their actual period to meet realistic requirements. For the *MP* flows, the period and packet length are randomly selected from the sets $\{4, 6, 8, 10, 12, 14, 16, 20\}$ ms and $\{1.5, 2, 2.5, 3, 3.5, 4, 4.5\}$ kB, respectively. The deadline for *MP* flows is selected as a random integer in the range $\{0.5 \times HP.p, HP.p\}$. The packet length of the burst flow is randomly selected from $\{0.6, 0.8, 1, 1.2\}$ kB. We define a super period as the simulation running timeline. Burst flows are randomly generated within this super period, which represents the time for burst flows to arrive; the deadline for burst flows is 1 ms. Table 3 shows the corresponding simulation parameters.

Table 3. Periodic flow division.

Parameter	Value	Parameter	Value	Parameter	Value
MTU	1500 B	ws	$6 \times MTU$	N_{in}	200
N_{out}	200	d_j^A	1 ms	B	1000 Mb/s
Batch	64	N_h	1024	γ	0.99
ϵ	0.9	ϵ_{decay}	0.995		
Parameter	Value	Parameter	Value		
HP.size	{0.6, 0.7, 0.8, 0.9, 1} kB	MP.size	{1.5, 2, 2.5, ..., 4.5} kB		
AP.size	{0.6, 0.8, 1, 1.2} kB	HP.period	{0.6, 0.8, 1, 1.2, 1.6} ms		
MP.period	{4, 6, 8, 10, 12, 14, 16, 20} ms				

For the configuration of the DQN network, the parameter settings are based on the common parameters of neural network design. We use three fully connected hidden layers deployed together with the input and output layers. The size of the hidden layer is 1024. We set the maximum deadline time slot length to 200 as the input and output size. The batch size is 64, the discount factor is 0.99, the learning rate is 0.001, the epsilon of the random greedy algorithm is set to 0.9, the drop parameter is 0.995, and the reward weights r_1 and r_2 are set to 0.3 and 0.7.

5.2. Baselines and Evaluation Metrics

To comprehensively evaluate the effectiveness of the proposed A-TSN framework, we compare it against several state-of-the-art scheduling algorithms for time-sensitive networking. The baseline methods are selected to cover both traditional heuristic-based scheduling for periodic flows and recent deterministic approaches for burst flows.

1. HSTCS : HSTCS integrates the TAS and CQF mechanisms for periodic flow scheduling. Specifically, it employs TAS for high-priority flows and CQF combined with an enhanced simulated annealing algorithm to determine injection time slots for medium-priority flows [29].
2. D-TSN: D-TSN is a baseline scheduling framework designed for burst flow scheduling in TSN. It applies deterministic scheduling principles without deep reinforcement learning, utilizing static slot allocation and deadline-based prioritization to ensure end-to-end delay guarantees for burst flows [25].

The subsequent experimental results evaluate the proposed two-stage A-TSN framework in detail. Specifically, the DQN-DITS algorithm is compared against traditional periodic flow scheduling baselines to demonstrate its advantages in scheduling success rate and latency reduction for static periodic flows. Meanwhile, the DDO algorithm is evaluated against the D-TSN baseline to validate its effectiveness in scheduling dynamically arriving burst flows with strict deadline guarantees.

5.3. Results and Analysis

This section presents the experimental results evaluating the proposed A-TSN framework. We analyze the performance of DQN-DITS for static periodic flow scheduling and DDO for dynamic burst flow scheduling. Furthermore, we discuss the framework's robustness under flow variations and its scalability.

The DQN-DITS algorithm is designed to schedule multiple types of time-sensitive periodic flows. Within each hybrid scheduling period, a subsequent periodic flow is scheduled only after the preceding one has been injected and allocated to its time slot. Consequently, the flow sorting strategy plays a crucial role in determining the scheduling success rate and the maximum number of schedulable flows.

To evaluate this impact, four sorting methods were compared, as illustrated in Figure 4. Periodic flows were sorted based on (i) packet length (descending order), (ii) period (ascending order), (iii) deadline, and (iv) randomness. The results indicate that sorting flows by packet length in descending order enables the scheduling of the largest number of MP flows, substantially outperforming sorting by period. This superiority arises because assigning smaller packet flows earlier preserves bandwidth availability and buffer capacity, thereby enhancing injection slot flexibility and enabling more flows to be scheduled within the period constraints.

Furthermore, we compare DQN-DITS with state-of-the-art periodic flow scheduling mechanisms. The HSTCS approach integrates TAS and CQF, where TAS handles *HP* flows, while CQF, combined with an enhanced simulated annealing algorithm, schedules *MP* flows. In contrast, DQN-DITS employs a Deep Q-Network to allocate injection time slot offsets dynamically within each period.

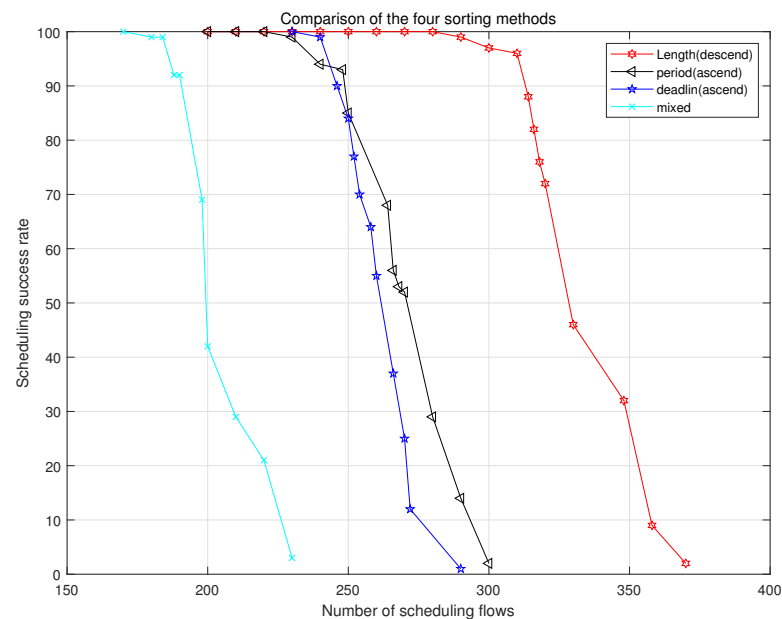


Figure 4. Comparison of different flow ordering methods on the maximum number of schedulable MP flows in the DQN-DITS algorithm.

As shown in Figure 5, DQN-DITS significantly outperforms HSTCS, achieving up to 300 schedulable MP flows, representing a 160% improvement over HSTCS. This performance gain is attributed to the DQN's learning-based slot allocation strategy, which adapts to flow characteristics and network states in real time to optimize scheduling decisions.

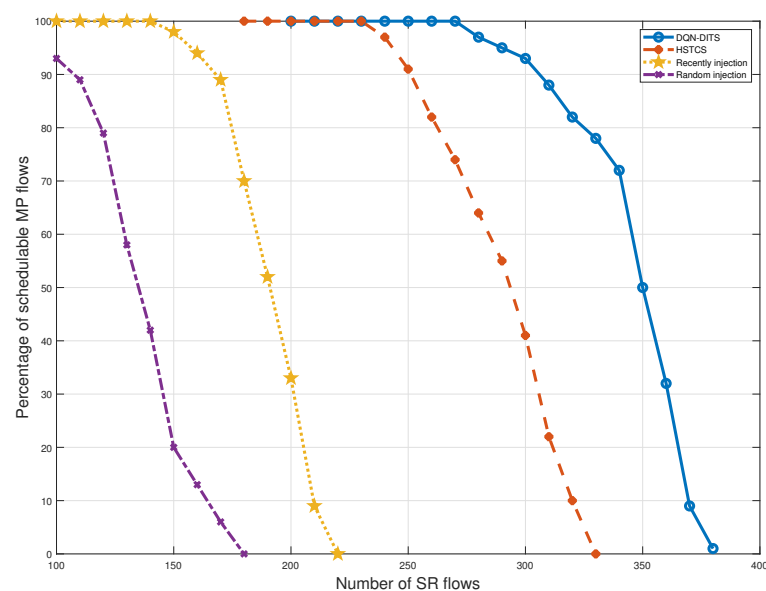


Figure 5. Comparison of the maximum number of schedulable MP flows between different injection methods under different network loads.

Additionally, we compared DQN-DITS with HSTCS, FITS, and Tabu search algorithms under identical network configurations and flow parameters. The experiments, repeated 100 times on a simulated network with 10HP flows and 260–290MP flows, yielded the results shown in Figure 6. DQN-DITS achieved a consistent 100% scheduling success rate across all trials.

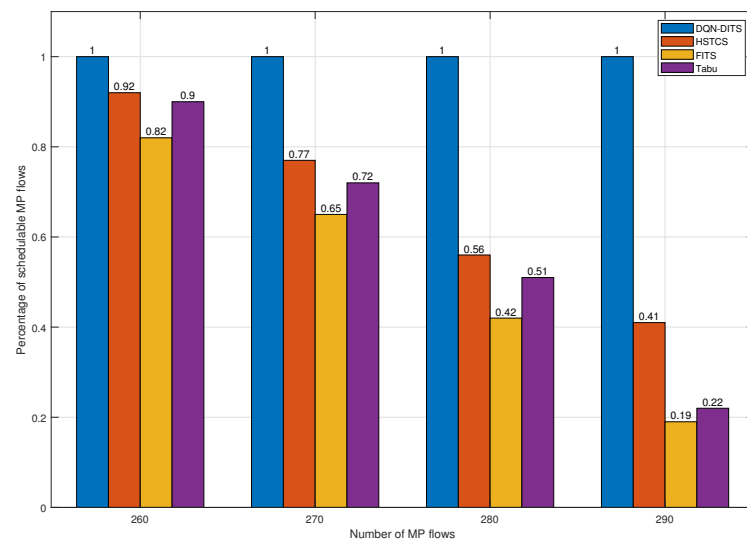


Figure 6. Scheduling success rates of different algorithms in mixed periodic flows.

These results demonstrate that the proposed DQN-DITS algorithm effectively maximizes the number of schedulable periodic flows while ensuring zero scheduling failures. Its reinforcement-learning-based approach dynamically adapts injection slot allocations to varying traffic demands, outperforming heuristic or optimization-based baselines in both success rate and scalability. Moreover, its low computational complexity renders it suitable for real-time TSN scheduling applications.

This section evaluates the performance of the proposed DDO algorithm compared to the latest burst flow scheduling algorithm, D-TSN. The experiments simulate burst flow scheduling both before and after periodic flows are processed by DQN-DITS.

As illustrated in Figure 7, when the number of MP flows is around 300, the DDO algorithm maintains nearly a 100% scheduling success rate. As the number of MP flows increases further, the scheduling success rates of both algorithms decrease; however, DDO consistently outperforms D-TSN, achieving approximately a 15% higher success rate under high load conditions. Furthermore, when combined with DQN-DITS pre-processing, the DDO algorithm maintains a performance improvement of nearly 10% in the proportion of schedulable flows compared to D-TSN, as shown in Figure 7.

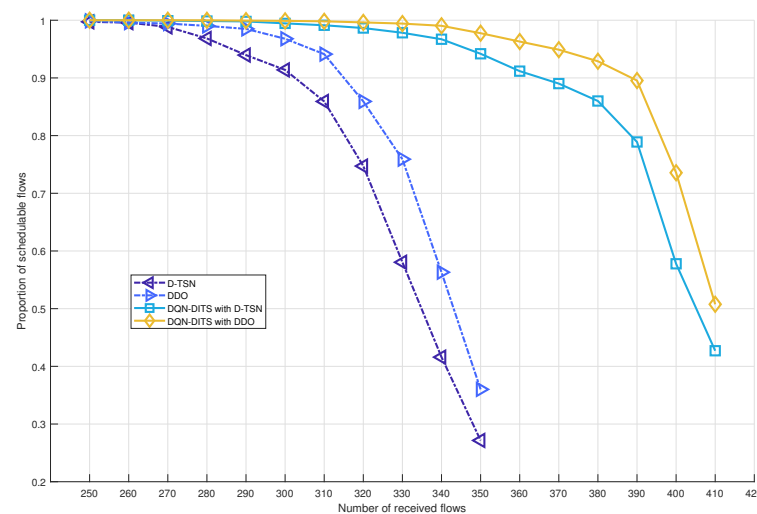


Figure 7. Comparison of schedulable burst flow ratios under DDO and D-TSN algorithms.

We also compare the scheduling delay performance of DDO and D-TSN. Figures 8 and 9 show the average time slot delay of MP flows scheduled by the two algorithms. After DQN-DITS processing, the DDO algorithm reduces the average scheduling delay of MP flows by approximately 40 time slots compared to D-TSN. This improvement is attributed to the maximum waiting delay constraint enforced during the DDO queuing process, which limits queuing latency and ensures timely flow transmission.

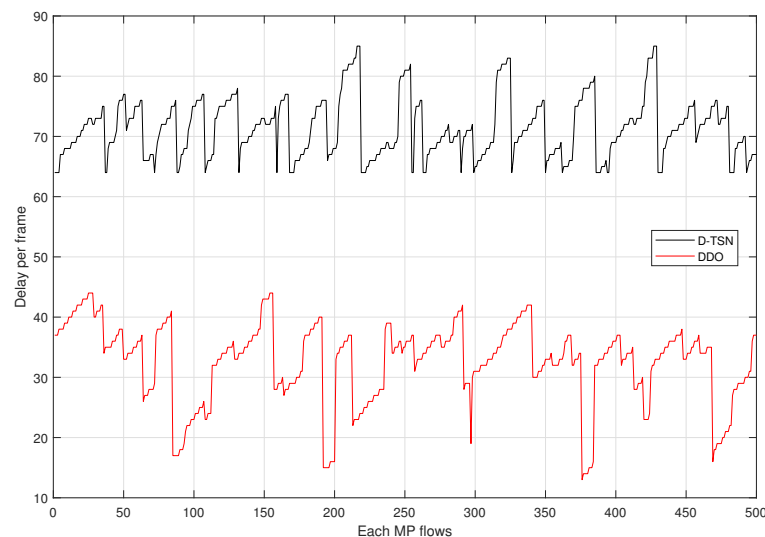


Figure 8. Average scheduling latency with DQN-DITS pre-processing.

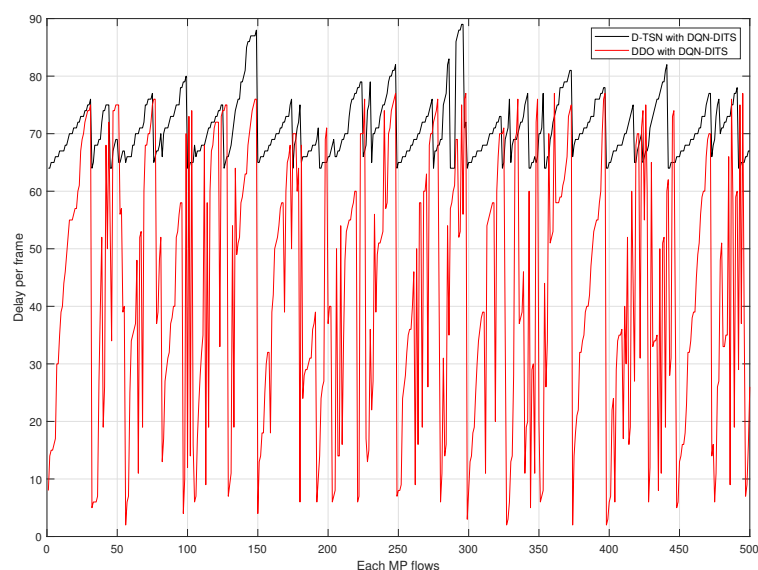


Figure 9. Average scheduling latency of without DQN-DITS pre-processing.

The superior performance of DDO arises from its dynamic deadline adjustment mechanism, which prioritizes burst flows based on both their urgency and the current network state. Unlike D-TSN's static slot allocation, DDO adaptively reallocates slots in real time, ensuring more flows are scheduled within their deadlines while reducing scheduling delays. Overall, these results demonstrate that DDO effectively enhances both the schedulability and timeliness of burst flows in mixed TSN traffic environments.

6. Conclusions

This paper proposes a two-stage asymmetric flow scheduling framework with dynamic deadline control, termed A-TSN, to address the challenges of deterministic transmission in asymmetric industrial network environments. In the first stage, the DQN-DITS algorithm leverages deep reinforcement learning to optimize injection slot allocation for periodic flows based on their deadline requirements. In the second stage, the DDO algorithm adaptively schedules both periodic and burst flows by dynamically updating their transmission priorities. Simulation results validate that the proposed architecture effectively achieves low-latency hybrid flow transmission, increases the successful scheduling rate of periodic flows, and ensures timely, wait-free delivery of burst traffic. These results highlight the potential of A-TSN in advancing deterministic communication for Industry 5.0 applications and provide a foundation for future research in intelligent TSN scheduling under mixed-traffic scenarios. In future work, we aim to explore the scalability and adaptability of the A-TSN framework in larger and more dynamic industrial network environments. Further research will focus on enhancing the hybrid scheduling mechanism to accommodate evolving traffic patterns and real-time network conditions. Additionally, we plan to investigate the integration of machine learning and optimization techniques to improve the performance and efficiency of TSN-based systems in Industry 5.0 applications.

Author Contributions: Conceptualization: H.D., T.Z., L.W. and F.L.; Methodology: H.D. and T.Z.; Software: H.D.; Validation: H.D. and T.Z.; Formal analysis: C.L.; Investigation: C.L.; Writing—original draft preparation: T.Z.; Writing—review and editing: H.D., T.Z., L.W. and F.L.; Supervision: F.L.; Project administration: F.L.; Funding acquisition: F.L. and H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant 62371249; the Key Project of Natural Science Foundation of Jiangsu Province under Grant BE2023087; and the Open Research Fund of Jiangsu Engineering Research Center of Communication and Network Technology, NJUPT, JSGCZX23010.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: We also acknowledge the editor for their valuable suggestions.

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

References

1. Kadhim, J.Q.; Aljazeera, I.A.; ALRikabi, H.T.S. Enhancement of online education in engineering college based on mobile wireless communication networks and IoT. *Int. J. Emerg. Technol. Learn.* **2023**, *18*, 176. [\[CrossRef\]](#)
2. Yu, H.; Taleb, T.; Zhang, J. Deep reinforcement learning-based deterministic routing and scheduling for mixed-criticality flows. *IEEE Trans. Ind. Inform.* **2022**, *19*, 8806–8816. [\[CrossRef\]](#)
3. Chen, M.; Zhao, L.; Chen, J.; Wei, X.; Guizani, M. Modal-aware resource allocation for cross-modal collaborative communication in IIoT. *IEEE Internet Things J.* **2023**, *10*, 14952–14964. [\[CrossRef\]](#)
4. Peng, Y.; Jolfaei, A.; Yu, K. A novel real-time deterministic scheduling mechanism in industrial cyber-physical systems for energy internet. *IEEE Trans. Ind. Inform.* **2021**, *18*, 5670–5680. [\[CrossRef\]](#)
5. Wu, B.; Wang, S.; Wang, J.; Tan, W.; Liu, Y. Flexible design on deterministic IP networking for mixed traffic transmission. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 4360–4365.
6. Xu, X.; Xu, Y.; Dou, H.; Chen, M.; Wang, L. Federated KD-Assisted Image Semantic Communication in IoT Edge Learning. *IEEE Internet Things J.* **2024**, *11*, 34215–34228. [\[CrossRef\]](#)
7. Zambouri, K.; Noor-A-Rahim, M.; John, J.; Sreenan, C.J.; Poor, H.V.; Pesch, D. A comprehensive survey of wireless time-sensitive networking (tsn): Architecture, technologies, applications, and open issues. *IEEE Commun. Surv. Tutor.* **2024**, *60*, 1. [\[CrossRef\]](#)
8. Xie, G.; Xiao, X.; Liu, H.; Li, R.; Chang, W. Robust time-sensitive networking with delay bound analyses. In Proceedings of the 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021; pp. 1–9.

9. Wang, D.; Jin, X.; Feng, Z.; Deng, Q. B-UFS: Uniform Resource Metric-Based Periodic Flow Scheduling in 5G-TSN Integrated Network. In Proceedings of the 2024 IEEE 14th International Symposium on Industrial Embedded Systems (SIES), Chengdu, China, 23–25 October 2024; pp. 140–147.
10. Yan, W.; Wei, D.; Fu, B.; Li, R.; Xie, G. A mixed-criticality traffic scheduler with mitigating congestion for CAN-to-TSN gateway. *Acm Trans. Des. Autom. Electron. Syst.* **2024**, *29*, 1–28. [[CrossRef](#)]
11. Zhu, J.; Zong, C.; Guo, W.; Liu, M.; Zhao, C. Exploring Deterministic Transmission Optimization in TSN. In Proceedings of the 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Nanjing, China, 29–31 March 2024; pp. 1292–1297.
12. Bura, A.; Bobbili, S.C.; Rameshkumar, S.; Rengarajan, D.; Kalathil, D.; Shakkottai, S. Structured Reinforcement Learning for Media Streaming at the Wireless Edge. In Proceedings of the 25th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '24), Athens, Greece, 14–17 October 2024; ACM: New York, NY, USA, 2024; MobiHoc '24, pp. 101–110.
13. Yang, Z.; Nguyen, P.; Jin, H.; Nahrstedt, K. MIRAS: Model-based Reinforcement Learning for Microservice Resource Allocation over Scientific Workflows. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 122–132.
14. Xu, Z.; Tang, J.; Meng, J.; Zhang, W.; Wang, Y.; Liu, C.H.; Yang, D. Experience-driven Networking: A Deep Reinforcement Learning based Approach. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 1871–1879.
15. Yoshimura, A.; Ito, Y. A Study on Determination of an Appropriate GCL of Time-Aware Shaper in Ethernet-Based Industrial Networks. In Proceedings of the 2024 Fifteenth International Conference on Ubiquitous and Future Networks (ICUFN), Budapest, Hungary, 2–5 July 2024; pp. 355–359.
16. Stüber, T.; Osswald, L.; Lindner, S.; Menth, M. A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN). *IEEE Access* **2023**, *11*, 61192–61233. [[CrossRef](#)]
17. Huang, Y.; Wang, S.; Li, G.; Zhang, X.; Xu, D.; Huang, T. Multi-path CQF for low-jitter and high-reliable packet delivery in time-sensitive networks. In Proceedings of the 2024 IEEE Wireless Communications and Networking Conference (WCNC), Dubai, United Arab Emirates, 21–24 April 2024; pp. 1–6.
18. Yin, C.; Li, Y.; Zhu, H.; He, X.; Han, W. HSTC: Hybrid traffic scheduling mechanism in time-sensitive networking. *J. Commun.* **2022**, *43*, 119–132.
19. Yan, J.; Quan, W.; Jiang, X.; Sun, Z. Injection time planning: Making CQF practical in time-sensitive networking. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 616–625.
20. Yu, S.; He, F.; Xie, A.; Zhao, L. Efficient Adaptive Bandwidth Allocation for Deadline-Aware Online Admission Control in Time-Sensitive Networking. *arXiv* **2025**, arXiv:2503.09093.
21. Huang, Y.; Wang, S.; Wu, B.; Huang, T.; Liu, Y. TACQ: Enabling zero-jitter for cyclic-queueing and forwarding in time-sensitive networks. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
22. Zhang, W.; Tang, N.; Zhang, C.; Guo, R.; Li, M.; Ying, C.; Jin, J. Two-Stage Resource Scheduling for Deterministic Communication and Computation Integration. In Proceedings of the GLOBECOM 2024-2024 IEEE Global Communications Conference, Cape Town, South Africa, 8–12 December 2024; pp. 4179–4184.
23. Dai, Q.; Chen, H.; Ni, Z.; Yang, Y.; Chen, L.; Fan, X.; Li, X. Reinforcement Learning-Based Resource Reservation for Mobile Edge Computing with Probable Failure. *IEEE Trans. Veh. Technol.* **2025**, *74*, 7985–7996. [[CrossRef](#)]
24. Cheng, Z.; Yang, D.; Zhang, W.; Ren, J.; Wang, H.; Zhang, H. DeepCQF: Making CQF scheduling more intelligent and practicable. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 1–6.
25. 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](#)]
26. Zhou, X.; He, F.; Zhao, L.; Li, E. Hybrid scheduling of tasks and messages for TSN-based avionics systems. *IEEE Trans. Ind. Inform.* **2023**, *20*, 1081–1092. [[CrossRef](#)]
27. Leonardi, L.; Lo Bello, L.; Patti, G. Combining earliest deadline first scheduling with scheduled traffic support in automotive TSN-based networks. *Appl. Syst. Innov.* **2022**, *5*, 125. [[CrossRef](#)]
28. Sánchez, J.C. Develop an algorithm to the flow allocation in asynchronous TSN network for the Industrial 4.0. *IEEE Trans. Ind. Inform.* **2024**, *28*, 1234–1245.
29. Li, Y.; Han, W.; Zhu, H.; Yin, C. A hybrid traffic scheduling mechanism applied to large scale time-sensitive networking. *Chin. J. Internet Things* **2023**, *7*, 72–87.

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.