Enhancing End-to-End Delay Satisfiability for IEEE 802.1Qav

Cheng Long, Zonghui Li, *Member, IEEE*, Shuxian Jin, Kang G. Shin, *Life Fellow, IEEE*, Daquan Zhang, Zixiao Wang, and Zhibo Pang, *Senior Member, IEEE*

***Abstract*—IEEE standard 802.1Qav evolves from Ethernet Audio Video Bridging (AVB) to support real-time transmission by performing Credit-Based Shaping (CBS) of flows at each hop. Prior work primarily focuses on achieving a tighter upper bound of the End-to-End (E2E) delay using network calculus, given the CBS configuration of flows. However, it is important to adjust the CBS configuration at each hop to meet the deadlines for various data flows. To this end, we address the CBS scheduling problem to enhance the satisfiability of E2E delay requirements. First, we formulate a linear constraint model and propose a Double Deep Q-Network (DDQN) reinforcement learning method, called `CBS-DDQN`, to solve the model. Second, to improve model training and scheduling, we present a fast method using equivalence reduction to calculate the E2E upper bounds of data flows for the evaluation of E2E delay satisfiability. Finally, we conduct an extensive evaluation of `CBS-DDQN`, demonstrating its ability to efficiently schedule thousands of flows and increase the number of flows meeting the delay requirements by 18–26% over prior methods.**

***Index Terms*—Credit-based shaping (CBS), reinforcement learning, AVB, time-sensitive networking (TSN).**

I. Introduction

OVER the past decade, digital transformation has increased the demand for highly intelligent, networked, and flexible time-sensitive systems. Evolving from traditional Ethernet, Time-Sensitive Networking (TSN) has emerged as a promising communication framework standardized by the IEEE 802.1 TSN Group [1]. TSN enables real-time and deterministic Ethernet-based communication, offering low packet loss and guaranteed delay bounds. It is particularly well-suited to meet the requirements of time-sensitive networking applications across multiple domains, including industrial automation, automotive networks, and power systems [2], [3], [4], [5].

To achieve real-time communication, TSN employs several traffic control mechanisms. IEEE 802.1Qav [6] enables asynchronous real-time transmission through the Credit-Based Shaper (CBS). It complements IEEE 802.1Qbv [7], which supports deterministic time-triggered transmission in fully synchronized networks. Unlike time-triggered mechanisms, CBS does not prescribe exact transmission instants for individual frames. Instead, under a given configuration and without requiring global time synchronization, CBS can provide deterministic upper bounds on the end-to-end delay of AVB flows. This property provides greater flexibility in system design and allows TSN to support a broader range of time-sensitive applications. Since IEEE 802.1Qav originates from Audio Video Bridging (AVB) technology, which was initially designed to guarantee real-time transmission of multimedia traffic, data flows utilizing CBS are commonly referred to as AVB flows.

Initially, CBS regulates two AVB queues, namely class A and class B. Class A has a higher priority than Class B. AVB flows are mapped into class A or class B queues. Each AVB queue uses a send slope ($sdSl$) and an idle slope ($idSl$) to control the increase and decrease of credits. They start with an initial credit value of zero. An AVB flow is permitted to transmit only when the credit of its corresponding class is greater than zero and no higher-priority flows are transmitting. The credit decreases at $sdSl$ when transmitting flows, and increases at $idSl$ when flows are waiting due to higher-priority traffic or negative credit. So, CBS regulates traffic transmission by dynamically adjusting credit values to prevent continuous bursts and thus ensure real-time transmission. The IEEE 802.1Q-2018 standard [8] extends CBS to support any number of AVB traffic classes, typically 8 AVB classes for eight queues. The higher the priority, the smaller the queue number is. In this paper, we assume that a smaller queue index corresponds to a higher priority for simplicity. This convention is adopted purely for notation and does not affect the generality of the proposed method.

Existing methods on CBS primarily focus on achieving tighter upper bounds for the E2E delay using Network Calculus (NC) [9]. Several studies [10], [11], [12], [13], [14] established NC-based models of CBS and derived worst-case delay (WCD) bounds. Other studies [15], [16], [17], [18] analyzed the relationship between reserved bandwidth ($sdSl$ and $idSl$) and WCD, and proposed methods for minimizing reserved bandwidth under

Received 11 November 2025; revised 12 March 2026 and 13 April 2026; accepted 18 May 2026. Date of publication 25 May 2026; date of current version 5 June 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62531003 and Grant 62272034 and in part by the Science and Technology Project of State Grid Corporation of China under Grant 52120025003R-295-LZ. *(Corresponding author: Zonghui Li.)*

Cheng Long, Zonghui Li, Shuxian Jin, Daquan Zhang, and Zixiao Wang are with the School of Computer Science & Technology, Beijing Jiaotong University, Beijing 100044, China (e-mail: long.c@bjtu.edu.cn; lizonghui@bjtu.edu.cn; 22125186@bjtu.edu.cn; dazhang@bjtu.edu.cn; w_zixiao@bjtu.edu.cn).

Kang G. Shin is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor MI 48109 USA (e-mail: kgshin@umich.edu).

Zhibo Pang is with the School of Advanced Manufacturing and Robotics, Peking University, Beijing 100871, China (e-mail: zhibo.pang@pku.edu.cn).

Digital Object Identifier 10.1109/TICPS.2026.3696600

E2E delay constraints. In addition, several studies [19], [20], [21] investigated load-balanced routing strategies on redundant networks to further reduce WCD. Despite these advances, most existing works focus on delay analysis under a predetermined mapping from flows to AVB classes across all network nodes. That is, if an AVB flow is predetermined as class B, the flow will be queued into class B in each hop of its routes. Under such assumptions, NC serves primarily as an analytical tool to evaluate the delay bound of a given configuration rather than to determine the configuration itself.

However, determining the class assignment of flows across network hops is inherently a combinatorial scheduling problem. The key idea of this work is to allow a flow to use different CBS classes at different hops along its route. Compared with a fixed end-to-end class mapping, dynamic per-hop assignment enables the scheduler to allocate higher priority only on the hops that are critical to satisfying the delay bound, while keeping lower priority on other hops to reduce interference to competing flows. From a theoretical perspective, this flexibility enlarges the feasible solution space of CBS configurations and thus increases the likelihood of finding schedules that satisfy heterogeneous deadline requirements.

Furthermore, it is not sufficient for flows within the same AVB class to share identical E2E delay bounds derived from these prior works. Even within the same class of AVB flows, different flows may have diverse deadline requirements. An AVB flow meets its real-time requirement only if its WCD is smaller than its deadline. We define the CBS configuration for the whole network as all maps from each AVB flow per hop to CBS classes. Therefore, determining the per-hop class assignment of AVB flows to satisfy heterogeneous deadline requirements remains a challenging and largely unexplored scheduling problem. This motivates the need for a scheduling framework that can determine feasible CBS configurations while ensuring the deadline requirements of diverse AVB flows.

To address this, we propose a novel approach, `CBS-DDQN`, which effectively schedules CBS configuration, namely the per-hop flow-to-priority mappings in CBS, to meet diverse deadline requirements of AVB flows. We make the following contributions:

1) Formulation of a linear constraint model for the CBS configuration to guarantee real-time transmission of AVB flows;
2) Proposal of a Double Deep Q-Network (DDQN) reinforcement learning method, called `CBS-DDQN`, to solve the scheduling problem;
3) Derivation of a fast method by equivalence reduction to calculate the E2E upper bounds of data flows for latency satisfiability evaluation;
4) Comparative evaluation on typical topologies to demonstrate that `CBS-DDQN` efficiently schedules thousands of AVB flows and increases the satisfiable number by 18–26% compared to existing methods.

The rest of the paper is organized as follows. Related work is reviewed in Section II. The system model and WCD analysis are presented in Section III. The CBS scheduling problem and `CBS-DDQN` are detailed in Section IV. Evaluation results and discussions are provided in Section V, followed by the conclusion in Section VI.

II. Related Work

IEEE 802.1Qav employs CBS to ensure the real-time transmission of AVB flows. The parameters of CBS include the number of CBS class queues, and each queue has two parameters, namely *sdSl* and *idSl*. The CBS configuration consists of mappings from each AVB flow at each hop to a specific CBS class. Both the parameters and the configuration of CBS cooperatively decide the E2E delay of AVB flows. Prior works of CBS typically focus on WCD analysis to achieve tighter E2E delay upper bounds of AVB flows.

Initially, some studies [10], [11] established the WCD analysis framework of two-class CBS using NC [9]. Besides NC, the compositional performance analysis (CPA) was also used to formally derive upper bounds on WCD of CBS in [12], [13], [14], [22]. But CPA is essentially the worst-case response time analysis for tasks on priority-based systems, and not for network transmission. When deadlocks of tasks happen, the WCD analysis will fail. NC is a more natural WCD analysis method for network flows by computing their service curves and arrival curves. Previous work [23] derived the service curves of class A and class B while considering the impacts of other traffic. Another study [24] extended the NC model of two classes to support multi-class CBS, while other studies [25], [26] tightened the WCD bounds by leveraging packet-level arrival curves instead of bit-level arrival curves. To address the WCD analysis of dynamic scenarios, previous work [27] proposed an incremental WCD analysis method to reduce the cost of re-computing WCD during traffic updates.

These methods compute the WCDs of AVB flows based on the predetermined parameters and configuration of CBS. That is, the same class queue has the same parameters, and a flow has the same CBS class across the whole network. As a result, the granularity of the WCD analysis is not a single flow but a class of flows. Flows belonging to the same class will have the same WCD as long as their routes are the same. In fact, each flow may have different deadlines. So, such WCD analysis methods weaken the E2E latency satisfiability of flow deadlines.

Some works were turned on the settings of CBS parameters. One study [28] analyzed the reserved-bandwidth impacts on WCD for a single switch by changing the per-class parameters (*sdSl* and *idSl*). Other studies [17], [18] proposed an allocation-and-evaluation feedback method to optimize the reserved-bandwidth parameter settings. Another study [15] reduced the cost of re-evaluation per allocation by local deadline decomposition. Previous work [16] provided a bandwidth reservation analysis for the schedulability of AVB flows and showed that excessive reserved bandwidth did not necessarily reduce the delay bound. In addition, other works balanced the traffic load by redundant routes [19], [20] and load-balanced routing [21], [29] to decrease the burst frame length per link, and thus reduce the WCD.

To the best of our knowledge, this paper is the first work to enhance E2E latency satisfiability for AVB flows by scheduling

the CBS configuration, namely all maps from each AVB flow per hop to CBS classes. We formulate such a scheduling problem as a linear constraint model including bandwidth reservation constraint, queue capacity constraint, and E2E delay constraint. And then, we maximize the number of satisfiable flows to achieve the CBS configuration.

III. Preliminary

This section first presents the background for the CBS traffic and network model, including the basic terminology and concepts, and WCD analysis for AVB flows.

A. Basic Terminology and Concepts

The network topology is modeled as a directed graph $G(V, E)$, where V denotes the set of terminal and switch nodes, and E denotes the set of directed edges between nodes. For each physical link between two nodes v_i and v_j, there are two corresponding directed data links in opposite directions, $(v_i, v_j) \in E$ and $(v_j, v_i) \in E$. Each directed edge e_l corresponds to a unique combination of an output port $v_{src}^{e_l}$ and an input port $v_{dst}^{e_l}$. Therefore, a flow's route can be represented by the set of directed edges $(e_1, e_2, \ldots, e_m)$. In this work, we focus on AVB traffic shaped by CBS in TSN networks. The scheduled traffic consists of AVB flows with E2E deadline requirements. In the evaluation, these flows are instantiated as industrial AVB traffic generated by field devices. Interactions with other TSN mechanisms, such as the Time-Aware Shaper (IEEE 802.1Qbv), guard bands, and frame preemption, are not considered.

We use Q_{e_l} to denote the set of AVB queues at the output port $v_{src}^{e_l}$. Each AVB queue $q_i^{e_l}$ in Q_{e_l} is associated with priority i and is defined by the following parameters:

$$q_i^{e_l} = \{\text{buffer}, \text{idSl}, \text{sdSl}, \text{burstSum}, \text{rateSum}\},$$

where $q_i^{e_l}$.buffer is the buffer capacity, $q_i^{e_l}$.burstSum denotes the aggregated frame sizes of flows, and $q_i^{e_l}$.rateSum is the sum of rates of flows, all associated with AVB queue $q_i^{e_l}$.

Let F denote the set of all AVB flows in the network. For a network with n AVB flows, we have $F = \{f_1, f_2, \ldots, f_n\}$. The attributes of f_k include its route, average rate, frame size, E2E delay deadline, and priorities on its route. Formally, f_k can be represented as:

$$f_k = \{\text{route}, \text{rate}, \text{size}, \text{deadline}, \text{priorities}\}.$$

For AVB flow f_k, f_k.size denotes the maximum frame size of the flow, and f_k.rate is the frame size divided by the period of f_k. f_k.priorities denotes the vector of per-hop priority assignments and is determined by the scheduling process as follows:

$$f_k.\text{priorities} = \{p_k^{e_l} \mid \forall e_l \in f_k.\text{route}\}.$$

B. WCD Analysis for AVB Traffic

NC is a deterministic analytical framework for evaluating worst-case performance bounds in communication networks [9], [30], [31], [32] . In NC, an arrival curve upper-bounds the cumulative traffic that a flow can generate over time, while a service curve lower-bounds the service guaranteed by a network element.

The worst-case delay bound can then be derived as the horizontal deviation between the arrival curve and the service curve. In the multi-class CBS NC model, the upper bound of the single-hop delay for any AVB Class i flow at the output port of a directed edge e_l is determined by the maximum horizontal deviation between the arrival curve $\alpha_i^{e_l}(t)$ and the service curve $\beta_i^{e_l}(t)$ for AVB Class i traffic:

$$D_i^{e_l} = h(\alpha_i^{e_l}(t), \beta_i^{e_l}(t)),$$

Assume f_k represents the *k-th* AVB flow whose priority remains in class i throughout transmission. The WCD for f_k is computed by summing hop/link delays along its route from the source node to the destination node:

$$WCD(f_k) = \sum_{e_l \in f_k.\text{route}} (D_i^{e_l} + d_{tech}^{e_l}),$$

where $d_{tech}^{e_l}$ represents other fixed latencies, such as processing delay, link delay, etc.

According to previous work [23], [24], the arrival curve and service curve of AVB flows in TSN can be expressed as follows. At the output port of a directed edge e_l, the service curve for AVB Class i traffic is given by

$$\beta_i^{e_l}(t) = \frac{C \cdot idSl_i}{idSl_i - sdSl_i} \left[t - \frac{credit_i^{\max}}{idSl_i} \right]^+, \tag{1}$$

where C is the transmission rate of the physical link and the notation $[x]^+$ is equal to x if $x \geq 0$ and 0 otherwise. The maximum and minimum credit values for AVB Class i traffic, denoted as $credit_i^{\max}$ and $credit_i^{\min}$, are provided by (2) and (3), respectively.

$$credit_i^{\max} = idSl_i \frac{\sum_{j=1}^{i-1} credit_j^{\min} - l_{>i}^{\max}}{\sum_{j=1}^{i-1} idSl_j - C}, \tag{2}$$

$$credit_i^{\min} = sdSl_i \frac{l_i^{\max}}{C}, \tag{3}$$

where $l_i^{\max}$ denotes the maximum packet length for AVB Class i traffic, and $l_{>i}^{\max}$ denotes the maximum packet length for all AVB and other traffic with a lower priority than class i.

The arrival curve of f_k at the output port of e_l is denoted as $\alpha_{i,k}^{e_l}(t)$. The arrival curve for AVB Class i traffic at the output port of e_l is given by:

$$\begin{aligned} \alpha_i^{e_l}(t) &= \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} \alpha_{i,k}^{e_l}(t) \\ &= \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} (\sigma_{i,k}^{e_l} + \rho_{i,k}^{e_l} \cdot t), \end{aligned} \tag{4}$$

where F^{e_l} is the set of flows transmitted through e_l, and $p_k^{e_l}$ denotes the priority of f_k on e_l. $\sigma_{i,k}^{e_l}$ and $\rho_{i,k}^{e_l}$ are the burst length and rate of f_k at the output port of e_l. $\alpha_{i,k}^{e_l}(t)$ can be derived from the arrival curve at the previous edge e_{l-1} along the route of f_k [9]:

$$\begin{aligned} \alpha_{i,k}^{e_l}(t) &= \alpha_{i,k}^{e_{l-1}}(t + D_{i,k}^{e_{l-1}}) \\ &= \sigma_{i,k}^{e_{l-1}} + \rho_{i,k}^{e_{l-1}} \cdot (t + D_{i,k}^{e_{l-1}}). \end{aligned} \tag{5}$$

At the source port of e_1, $\sigma_{i,k}^{e_1}$ is the frame size f_k.size, and $\rho_{i,k}^{e_1}$ is the average rate f_k.rate.

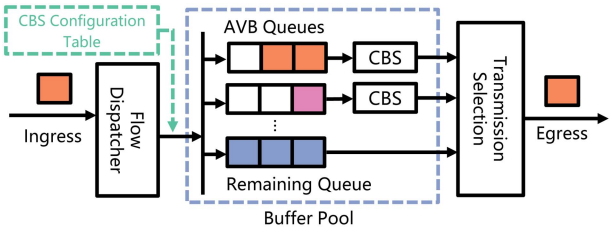


Fig. 1. The output port architecture in a TSN device featuring CBS.

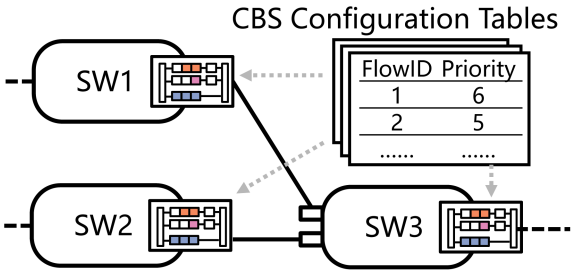


Fig. 2. An example of CBS configuration tables for output ports in TSN switches. These tables are the maps from *FlowID* to *Priority* queues of CBS.

IV. Proposed Solutions

This section first presents the CBS networking architecture. Based on such an architecture, we formalize the linear constraint model for CBS scheduling. Finally, we detail the proposed reinforcement learning scheduler to solve the scheduling problem.

A. CBS Networking Architecture

Fig. 1 illustrates the output port architecture of a TSN device (terminal or switch) featuring CBS. A flow enters the device from the ingress, passes through the dispatcher, and is stored in the Buffer Pool to wait for forwarding. The Buffer Pool consists of multiple AVB queues and a remaining queue. Each AVB queue is assigned a priority. The remaining queue is for other traffic. The dispatcher maps flows to the corresponding AVB queue or the remaining queue by looking for the CBS configuration table with their flow identification (FlowID). AVB flows are placed into AVB queues and transmitted after being shaped by CBS. At the Transmission Selection module, higher-priority AVB flows are transmitted first, while the remaining queue has the lowest priority.

Fig. 2 presents an example of CBS configuration tables for output ports in TSN switches. Each output port in a TSN device is associated with a CBS configuration table. The table contains two fields: FlowID and Priority. Each record in the table specifies the priority of a flow, identified by its FlowID, on the TSN device. The configuration tables are generated based on the proposed scheduler in this paper and are distributed to each device. Since each output port maintains its own FlowID-to-Priority mapping table, the same flow can be assigned different priorities at different hops along its route. During offline scheduling, the scheduler incrementally determines the per-hop priority assignments of flows along their routes. After the scheduling process completes, the CBS configuration tables of all output ports are generated from the final priority assignments of the admitted flows and installed on the corresponding TSN devices for packet dispatching.

CBS supports multiple priority queues, and each queue is associated with two parameter credit rates, namely $idSl_i$ and $sdSl_i$ for increasing and decreasing credits, respectively. The CBS scheduling problem is to decide the CBS priority queue of each AVB flow on each hop under the bandwidth constraints specified by the credit rate parameters.

Such a scheduling problem is computationally hard due to the combinatorial nature of per-hop class assignment under network-wide delay constraints. In the proposed RL-based framework, this complexity mainly manifests in the repeated NC delay analysis performed during the scheduling process, which becomes the dominant computational bottleneck. Its potential benefit comes from the ability to improve real-time guarantees through per-hop flow-to-class mappings, without requiring a flow to remain at the same high priority along the entire route. Compared with a fixed class assignment, the scheduling variable becomes a vector of per-hop class choices for each flow, which expands the feasible scheduling space but also increases the search complexity with both the number of flows and the hop count.

The proposed CBS-DDQN framework is designed to explore this expanded space efficiently, while the NC-based constraint checking ensures that all accepted configurations satisfy the analytical bandwidth, queue-capacity, and delay constraints. To support practical cases better, we consider eight priority queues in each hop according to IEEE 802.1Q. Seven high-priority queues are AVB queues, and the lowest priority queue is for other flows.

B. Linear Constraint Model

To satisfy the real-time transmission of different AVB flows, we formalize the linear constraints for the CBS configuration.

C_1: Bandwidth Reservation Constraint. In this work, $q_i^{e_l}$.idSl, $q_i^{e_l}$.sdSl, and the link rate C are all expressed in bps. Therefore, $q_i^{e_l}$.idSl directly represents the bandwidth reserved for queue $q_i^{e_l}$. According to IEEE 802.1Qav, the corresponding send slope satisfies $q_i^{e_l}.\text{sdSl} = q_i^{e_l}.\text{idSl} - C$. The reserved bandwidth of each queue, $q_i^{e_l}$, is specified by its credit accumulation rate, $q_i^{e_l}$.idSl. The reserved bandwidth should not be less than the sum of the arrival rates of AVB flows entering the queue, else AVB flows may experience frame drops. The constraint is expressed as:

$$\forall e_l \in E, \forall q_i^{e_l} \in Q_{e_l}:$$
$$\sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} f_k.\text{rate} \leq q_i^{e_l}.\text{idSl}.$$

C_2: Queue Capacity Constraint. The total frame size of AVB flows into each queue must not exceed the queue capacity, which can be expressed as:

$$\forall e_l \in E, \forall q_i^{e_l} \in Q_{e_l}:$$

$$\sum_{f_k \in F^{e_l} \wedge p_k^{e_l}=i} f_k.\text{size} \le q_i^{e_l}.\text{buffer}.$$

$\mathbf{C}_3$: E2E Delay Constraint. The WCD of each AVB flow calculated by NC must not exceed the deadline of the flow requirement, which is expressed as:

$$\forall f_k \in F:$$
$$WCD(f_k) \le f_k.\text{deadline},$$

where $WCD(f_k)$ is the WCD of f_k. Therefore, a flow is considered schedulable only when its NC-based worst-case delay bound does not exceed its deadline.

Optimization Objective: The objective of CBS scheduling is to maximize the number of flows that satisfy all constraints in $\{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$. This objective can be expressed as:

$$\max\left(\sum_{k=1}^{n} I_k\right), I_k = \begin{cases} 1, & f_k \text{ satisfies } \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3 \\ 0, & \text{otherwise.} \end{cases}$$

The linear constraint model for the CBS scheduling problem can be directly solved using a Satisfiability Modulo Theory (SMT) solver. The SMT solver searches for configurations satisfying the bandwidth reservation constraint ($\mathbf{C}_1$), queue capacity constraint ($\mathbf{C}_2$), and end-to-end delay constraint ($\mathbf{C}_3$). Since the objective of the scheduling problem is to maximize the number of schedulable flows, the SMT solver is used to determine feasible configurations under these constraints. However, SMT solvers cannot handle large-scale flows due to their exponential complexity and significant time cost.

C. Details of CBS-DDQN

We propose a DDQN reinforcement learning scheduler called `CBS-DDQN` for thousands of AVB flows. Fig. 3 illustrates the workflow of our priority scheduling framework. In `CBS-DDQN`, CBS scheduling is represented as an agent that considers observed state, selected action, and earned reward with the environment to enhance scheduling decisions.

1) State and Feature Extraction: The state s_t includes the information about the current flow and the allocation of resources. The current flow information includes the parameters of f_k, excluding f_k.priorities as it is not yet decided. (f_k.rate, f_k.size, f_k.deadline) are standardized into a raw vector x_t. The resource allocation is considered only for edges along f_k.route. Each link resource of e_l is considered as the link resource vector z_l, which comprises the transmission rate C, fixed delays $d_{tech}^{e_l}$, and data on all AVB queues Q_{e_l}:

$$z_l = C||d_{tech}^{e_l}||Q_{e_l},$$

where $||$ denotes the concatenation operator. Zero padding is applied based on the maximum number of hops M to ensure uniform dimensions. A multi-layer perceptron module is employed for feature extraction of the resource allocation information. The state s_t at time step t is expressed as

$$s_t = \mathcal{M}(z_1||z_2||\ldots||z_M)||x_t,$$

where $\mathcal{M}(\cdot)$ denotes the multi-layer perceptron module.

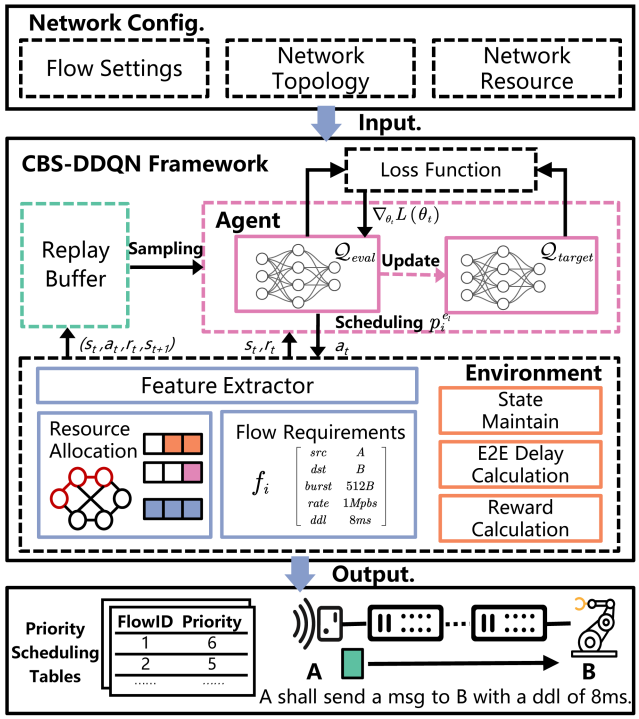


Fig. 3. Overview of `CBS-DDQN` Framework.

2) Action and State Transition: The agent takes s_t as input and generates an action a_t, which corresponds to f_k.priorities. We define a_t as follows.

$$a_t = [a_{t,1}^T, a_{t,2}^T, \ldots, a_{t,M}^T] \in \mathbb{R}^{M \times N},$$

where N is the number of AVB queues. The vector $a_{t,i} \in \mathbb{R}^N$ represents the decision of the AVB queue for f_k on e_i. Each element $a_{t,i,j} \in \{0, 1\}$ of $a_{t,i}$ indicates whether f_k is transmitted through the AVB queue of class j on edge e_i. Specifically, $a_{t,i,j} = 1$ means that $p_k^{e_i} = j$. Only one element in each $a_{t,i}$ can be "1". The ϵ-*greedy* exploration strategy is employed to balance exploration and exploitation. In each iteration, the agent selects a random action with probability ϵ or chooses the action with the highest $\mathcal{Q}$ value based on the current state s_t with probability $1 - \epsilon$.

Before scheduling starts, the candidate AVB flows are sorted by increasing frame size and average rate to define the admission order. Therefore, smaller and lower-bandwidth flows are considered earlier in the sequential scheduling process. Upon taking an action a_t, the environment transitions from state s_t to state s_{t+1} based on a_t. The state transition involves evaluating whether the priority assignment f_k.priorities determined by a_t satisfies the constraints $\{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$. Constraints $\mathbf{C}_1$ and $\mathbf{C}_2$ can be checked without altering the network; if either fails, f_k is considered unschedulable. In CBS, the scheduling of each flow affects the latency of all previously configured flows, necessitating a reevaluation of the E2E latency for all prior flows. Specifically, the environment forms a temporary scheduling result consisting of the current flow together with all previously admitted flows, and then recomputes the NC-based delay bounds under their current per-hop priority assignments. If f_k or any previously

scheduled flow fails to satisfy the constraint $\mathbf{C}_3$, then f_k is deemed unschedulable.

If f_k is successfully scheduled, the environment updates the AVB queues along f_k.route and transitions to scheduling the next flow in the sorted admission order. After all flows have been processed, the final per-switch CBS configuration tables are generated from the recorded per-hop priority assignments of the admitted flows. Otherwise, the environment's resources remain unchanged, the unschedulable flow f_k is mapped to the remaining queue for transmission, and the environment transitions to scheduling the next flow.

3) Reward and Model Training: The environment provides a reward based on a_t. The reward structure considers both the scheduling outcome and resource allocation efficiency. It also favors assigning higher priorities to flows with smaller frame sizes and lower bandwidth demands, because such flows consume fewer resources while being easier to admit early in the scheduling process. A queue with excessive bandwidth usage would be a bottleneck for scheduling. We define ϕ_t as the maximum occupied bandwidth, which serves as a penalty term in the reward function.

$$\phi_t = \max_{e_l \in E, 1 \le i \le N} \left(\frac{q_i^{e_l}.\text{rateSum}}{q_i^{e_l}.\text{idSl}} \right).$$

To encode the preference for small and low-bandwidth flows receiving higher priorities, we define

$$\psi_t = \left(1 - \frac{f_k.\text{size}}{s_{\max}}\right) \times \left(1 - \frac{f_k.\text{rate}}{r_{\max}}\right) \times \frac{1}{|f_k.\text{route}|} \times \sum_{e_l \in f_k.\text{route}} \frac{N - p_k^{e_l}}{N-1},$$

where $s_{\max}$ and $r_{\max}$ are the maximum frame size and average rate in the current training batch, respectively.

If the current flow f_k is successfully scheduled, the environment returns a reward of 1 plus the preference term and minus the bandwidth-occupancy penalty; otherwise, it returns 0. The reward function can be formulated as

$$r_t = \begin{cases} 1 + \mu\psi_t - \lambda\phi_t, & f_k \text{ satisfies } \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3 \\ 0, & \text{otherwise,} \end{cases}$$

where μ and λ are the weights for the preference term and the penalty term, respectively.

During each training epoch, the algorithm randomly selects a set of flows from the training set, sorts them by increasing frame size and average rate, and schedules them sequentially, yielding a set of experience samples (s_t, a_t, r_t, s_{t+1}) used for updating network parameters. DQN algorithms employ a single *Q-function* for both action selection and evaluation, which can overestimate the parameters [33]. To mitigate this problem, the Double DQN algorithm introduces two distinct *Q-functions:* the evaluation network ($\mathcal{Q}_{eval}$) and the target network ($\mathcal{Q}_{target}$). The target $\mathcal{Q}$ value, y_t, is computed using both $\mathcal{Q}_{eval}$ and $\mathcal{Q}_{target}$, where $\mathcal{Q}_{eval}$ selects the action, and $\mathcal{Q}_{target}$ provides the corresponding $\mathcal{Q}$ value. The target $\mathcal{Q}$ value y_t is given by

$$y_t = r_t + \gamma \mathcal{Q}(s_{t+1}, \underset{a \in A}{argmax}\, \mathcal{Q}(s_t, a; \theta); \theta^-),$$

Algorithm 1: Recursion-free WCD Calculation.

Input: Network: G, Scheduling flow set: F
Output: WCD set: WCD

```
Q ← empty FIFO queue
D_in ← empty array
{V', E'} ← CreatePreGraph(G, F)
Update D_in with indegree of V'
for e_l ∈ V' do
    if D_in[e_l] = 0 then Add e_l to Q
while Q is not empty do
    e_l ← Q.remove
    foreach priority i on edge e_l do
        σ_i^{e_l} ← 0
        foreach f_k ∈ F^{e_l} AND i = p_k^{e_l}, j = p_k^{e_l-1} do
            σ_i^{e_l} += σ_{j,k}^{e_l-1} + ρ_{j,k}^{e_l-1} · D_j^{e_l-1}
        φ_i ← idSl_i, ω_i ← credit_i^max / idSl_i
        D_i^{e_l} ← σ_i^{e_l} / φ_i + ω_i
    Update D_in, Q with indegree of V'
foreach f_k ∈ F do
    WCD[k] ← 0
    foreach e_l ∈ f_k.route AND i = p_k^{e_l} do
        WCD[k] += D_i^{e_l} + d_tech^{e_l}
return WCD
```

where γ is the discount factor, θ represents the parameters of the $\mathcal{Q}_{eval}$, and θ^- represents the parameters of $\mathcal{Q}_{target}$. The parameters of $\mathcal{Q}_{eval}$ are updated using stochastic gradient descent with the following loss function:

$$L(\theta) = \mathbb{E}[(y_t - \mathcal{Q}(s_t, a_t; \theta))^2].$$

At regular intervals, the parameters θ of $\mathcal{Q}_{eval}$ are copied to θ^- of $\mathcal{Q}_{target}$ to maintain stability in the target network. The prioritized experience replay [34] is implemented to emphasize important experience samples and accelerate model training.

D. Acceleration for CBS-DDQN

According to (4), allocating f_k affects the arrival curves of related flows at each node along its route, affecting their WCD and hence requiring its recalculation. According to (5), recursive computation at each hop via the NC model can cause a stack overflow at large data scales. Computing WCD for AVB flows is a major bottleneck in both model training and runtime for CBS priority scheduling, because the NC analysis must be repeatedly invoked to evaluate candidate scheduling decisions in the reinforcement learning loop. To enhance efficiency, we apply equivalence reduction to the multi-class AVB traffic NC model [24] and propose a rapid calculation method.

1) WCD Analysis Model: For AVB Class i traffic on e_l, let $\rho_i^{e_l} = \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} (\rho_{i,k}^{e_l})$ and $\sigma_i^{e_l} = \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} (\sigma_{i,k}^{e_l})$. The arrival curve (4) can then be expressed as:

$$\alpha_i^{e_l}(t) = \rho_i^{e_l} \cdot t + \sigma_i^{e_l}, t \ge 0.$$

The $sdSl_i$ is the difference between $idSl_i$ and the physical link rate C according to the IEEE 802.1Qav standard [6], i.e.,

$sdSl_i = idSl_i - C$. The service curve (1) can be expressed as:

$$\beta_i^{e_l}(t) = idSl_i \left[t - \frac{credit_i^{\max}}{idSl_i} \right]^+. \tag{6}$$

Let $\varphi_i = idSl_i$ and $\omega_i = \frac{credit_i^{\max}}{idSl_i}$, then (6) can be expressed as:

$$\beta_i^{e_l}(t) = \begin{cases} \varphi_i(t - \omega_i), & t \geq \omega_i \\ 0, & 0 \leq t < \omega_i. \end{cases}$$

According to constraint $\mathbf{C}_1$, we have $\rho_i^{e_l} \leq \varphi_i$. The maximum horizontal deviation $h(\alpha_i^{e_l}(t), \beta_i^{e_l}(t))$, representing an upper bound on the delay $D_i^{e_l}$ for AVB Class i flow on e_l, is obtained at $\alpha_i^{e_l}(t) = \beta_i^{e_l}(t + D_i^{e_l}) = \rho_i^{e_l}$. The value of $D_i^{e_l}$ can be determined as:

$$D_i^{e_l} = \frac{\sigma_i^{e_l}}{\varphi_i} + \omega_i.$$

2) Equivalence Reduction for Calculation: The delay upper bound for AVB Class i flow on e_l can be determined using $\sigma_i^{e_l}$, φ_i, and ω_i. While φ_i and ω_i are directly related to all AVB flows on e_l and can be directly calculated, $\sigma_i^{e_l}$ requires recursive calculation. According to (5), for directed edge e_l, $\sigma_i^{e_l}$ depends on the information of the previous directed edge e_{l-1} on the route of each AVB Class i flows:

$$\sigma_i^{e_l} = \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} (\sigma_{j,k}^{e_{l-1}} + \rho_{j,k}^{e_{l-1}} \cdot D_j^{e_{l-1}}),$$

$$\rho_i^{e_l} = \sum_{f_k \in F^{e_l} \wedge p_k^{e_l} = i} \rho_{j,k}^{e_{l-1}},$$

where j represents the priority of f_k on e_{l-1}, i.e., $j = p_k^{e_{l-1}}$.

To eliminate recursive calculations and enhance efficiency, topological sorting is employed to remove recursion, as detailed in Algorithm 1 and described below:

- Construct a directed graph based on the routes of scheduled AVB flows, where each edge $e_l \in E$ is a vertex. If there is a route $(\ldots, e_{l-1}, e_l, \ldots)$, then a directed edge exists between the corresponding vertices of e_{l-1} and e_l, indicating that the calculation of a delay upper bound on e_l depends on e_{l-1}.
- Perform topological sorting on the directed acyclic graph to generate the topological sequence. Repeatedly identify the vertex with an in-degree of zero, output it to the sequence, and remove it from the graph until all vertices are processed.
- The predecessors of each vertex in the topological sequence have already been calculated. Thus, calculating the delay for each class of AVB flows on each edge $e_l \in E$ enables recursion-free delay computation.

3) Computational Complexity: Let $|E|$ denote the number of directed edges in the network and $|F|$ denote the number of AVB flows. In the original recursive NC-based WCD computation, delay information is propagated hop by hop along the route of each flow, which may lead to repeated recursive evaluations of upstream dependencies. After equivalence reduction, we construct a route-dependency graph in which each network edge is treated as a vertex and dependency relations are introduced according to the flow routes. Topological sorting of this graph requires $O(|E|)$ time. During the traversal, the delay information associated with each flow is propagated along its route only once. Since each flow may traverse multiple edges along its path, the total propagation cost is proportional to the number of flows and the route lengths. Therefore, the overall complexity of one WCD recomputation is bounded by $O(|E| + |F| \cdot |E|)$ in the worst case. This converts the recursive delay analysis into a non-recursive traversal of the dependency graph and avoids repeated evaluations of upstream nodes, which significantly improves the efficiency of NC analysis within the reinforcement learning loop.

In many industrial TSN deployments, schedule synthesis is performed offline during system design or deployment and may only need to be computed once. In such scenarios, satisfying the timing requirements of AVB flows is more important than reducing the runtime of the scheduling algorithm. The primary objective of the proposed method is therefore to generate CBS configurations whose NC-based worst-case delay bounds do not exceed the flow deadlines. Computational efficiency is nevertheless relevant in practical system design, especially for large-scale networks or when multiple configuration scenarios must be evaluated during network planning or system updates. Hence, the proposed acceleration improves practicality while preserving the timing guarantees required by industrial applications.

V. Evaluation and Analysis

To evaluate the effectiveness of the proposed method, we compare `CBS-DDQN` with several baseline approaches under typical TSN network topologies. The compared methods are listed as follows:

- *CBS-DDQN:* The proposed reinforcement learning–based scheduling method that dynamically assigns AVB priorities for each flow at each hop.
- *CBS-SA:* A state-of-the-art scheduling method, termed Strategic Annealing Scheduling, proposed in [35], which assigns priorities under fixed CBS bandwidth reservations to balance scheduling performance.
- *Uniform-Small:* A heuristic strategy where flows with smaller packet sizes are assigned higher priorities.
- *Uniform-Large:* A heuristic strategy where flows with larger packet sizes are assigned higher priorities.
- *CBS-SMT:* A constraint-based approach that determines priorities using the Z3 SMT solver [36]. The SMT solver is used as an exact feasibility solver for small-scale instances, where it searches for configurations satisfying the constraints defined in the linear model.

The comparisons are conducted across various configurations within typical network topologies, using the shortest path algorithm [37] for routing in all methods.

We evaluate the scheduling performance using two metrics: (1) *schedulability*, defined as the proportion of flows whose worst-case end-to-end delay bounds satisfy their deadline constraints, and (2) *runtime*, which measures the computational cost required to obtain a feasible scheduling configuration.

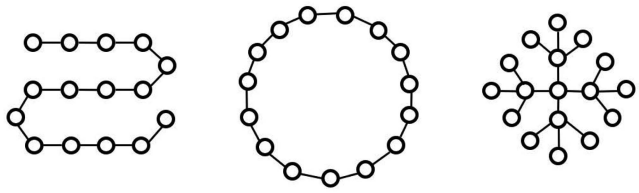


Fig. 4. Three typical industrial control network topologies.

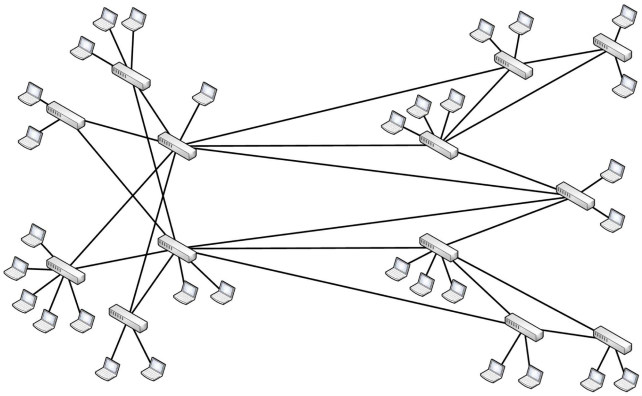


Fig. 5. The topology of the Orion CEV network.

A. Experimental Setup

1) Topology Selection: We select the following topologies to represent various application scenarios: **Typical Topologies** used in our comparative experiments include linear, ring, and snowflake structures, as shown in Fig. 4. **Orion Crew Exploration Vehicle** (CEV) [38] is utilized as a real-world network topology to validate the algorithm's effectiveness, as shown in Fig. 5. This topology employs a hybrid structure, integrating elements of star, tree, and ring structures. In our experiments, the maximum hop count is set to 6.

2) Flow Features: The AVB flows used in our experiments represent industrial AVB traffic generated by field devices and are generated with the following configuration. The parameter ranges are selected to reflect practical TSN deployments and are consistent with the IEC/IEEE 60802 profile [39] for industrial automation networks. (a) The sending and receiving terminals for each flow are randomly selected from all terminals in the network; (b) The frame size of each flow is randomly selected between 64 bytes and 1,518 bytes, matching the standard Ethernet frame size range; (c) The average rate of each flow is uniformly distributed between 0.5 Mbps and 1.5 Mbps, resulting in flow periods spanning both short-period (e.g., 1–4 ms) and long-period (e.g., 4–16 ms) regimes, thereby covering representative high-load and low-load conditions; (d) Under CBS priority scheduling, the E2E delay deadline for each flow is uniformly distributed between 4 ms and 32 ms, consistent with the latency requirements specified for AVB traffic in IEEE 802.1BA [40].

3) Resource Setting: The transmission bandwidth of physical links in the network topology is set to 1 Gbps. Each device port uses 8 priority queues. The buffer size for each queue is set to 1 MB, following a realistic configuration commonly adopted in TSN-related studies [41]. The higher-priority AVB queue has a higher $idSl$, i.e., $idSl_i > idSl_{i+1}$. The $idSl$ for each class i ($i = 1, 2, \ldots, 7$) is set to:

$$idSl_i = \frac{m - i + 1}{\sum_{k=1}^{m} k} C,$$

where m is the number of priority queues on each device port, which is set to 8 in this case. In the evaluation, the $idSl$ values are treated as predefined system parameters rather than adaptive variables determined by the aggregated queue state. This setting reflects a design workflow in which CBS bandwidth reservations are configured during system-level resource planning, and the scheduler then assigns flows to queues under the resulting bandwidth constraints.

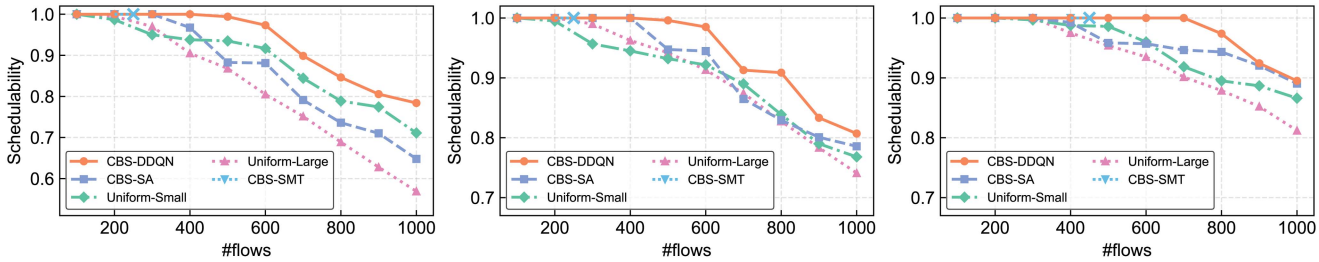
As a result, the evaluation isolates the impact of per-hop class assignment on schedulability under a fixed CBS bandwidth configuration and ensures a fair comparison across different scheduling strategies. In practical systems, when a new AVB flow is added to an already deployed network, adjusting the flow priority is often significantly easier than reconfiguring the reserved bandwidth parameters of all queues. Therefore, treating idle slopes as fixed parameters is a reasonable assumption for such operational scenarios.

We implement `CBS-DDQN` using PyTorch [42]. Both $\mathcal{Q}_{eval}$ and $\mathcal{Q}_{target}$ networks consist of two hidden layers with dimensions of 512 and 128, respectively. ReLU [43] is used as the activation function, and Adam [44] is employed as the optimizer for all modules. The initial learning rate is set to 10^{-4}, the discount factor is set to 0.99, and the weights of the preference term and penalty term are both set to 0.2. The experience replay buffer has a capacity of 20,000 transitions. Based on the above settings, we generated 3 batches with 1,000 flows per batch for training. The model is trained for 1,500 epochs over 3 days.

B. Performance Comparison Under Typical Network Scenarios

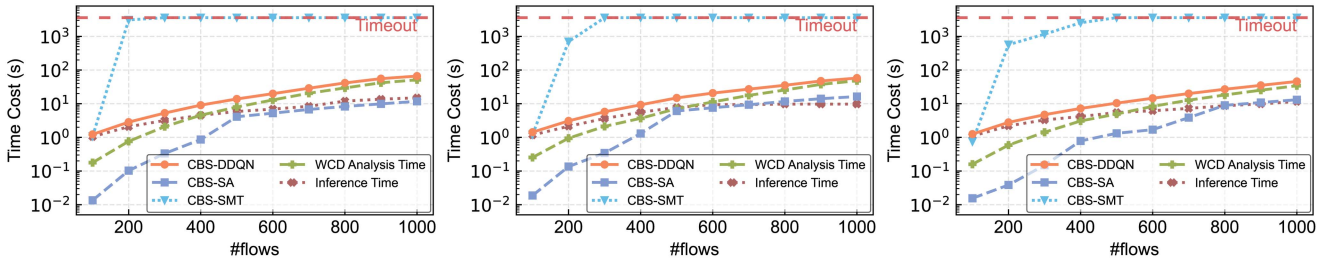
Figs. 6 and 7 provide an overview of `CBS-DDQN`'s performance in comparison with CBS-SA, Uniform-Small, and Uniform-Large on various typical topologies.

Figs. 6(a), 6(b), and 6(c) show the schedulability results. In the linear, ring, and snowflake topologies, `CBS-DDQN` consistently outperforms the other methods in terms of schedulability. For small-scale instances where SMT terminates, CBS-DDQN achieves the same number of schedulable flows. At 1,000 flows, the schedulability of `CBS-DDQN` reaches 0.784, 0.807, and 0.895 in the linear, ring, and snowflake topologies, respectively. These values are higher than those achieved by the baseline methods CBS-SA, Uniform-Small, and Uniform-Large. For example, in the linear topology, `CBS-DDQN` schedules approximately 7.2% more flows than CBS-SA, 7.3% more than Uniform-Small, and 21.5% more than Uniform-Large. Similar improvements can also be observed in the ring and snowflake topologies. In linear topology, all flows traverse a single path, leading to higher congestion on that route. In contrast, in ring and snowflake topologies, flows are distributed across multiple



(a) Schedulability on linear topology. (b) Schedulability on ring topology. (c) Schedulability on snowflake topology.

Fig. 6. Schedulability on different typical topologies.



(a) Time cost on linear topology. (b) Time cost on ring topology. (c) Time cost on snowflake topology.

Fig. 7. Time cost on different typical topologies.

paths, leading to a higher number of successfully scheduled flows compared to a linear topology.

Figs. 7(a), 7(b), and 7(c) illustrate the time costs on a logarithmic scale. The time costs for CBS-DDQN, CBS-SA, Uniform-Small, and Uniform-Large increase approximately linearly with the number of flows, while CBS-SMT's cost grows exponentially. CBS-SMT requires over an hour to process 200 to 400 flows, so its results for larger problems are not reported. When the number of flows becomes large or no feasible configuration exists, the SMT solver may continue exploring the constraint space until timeout, which limits its applicability to relatively small problem instances. In contrast, CBS-DDQN maintains stable runtime growth and can handle large-scale scenarios with up to 1,000 flows. With 1,000 flows, the runtime of CBS-DDQN is 66.2 s, 57.9 s, and 45.9 s for the linear, ring, and snowflake topologies, respectively. To further analyze the runtime components, we divide the total runtime of CBS-DDQN into two parts: the reinforcement learning inference time and the NC-based worst-case delay (WCD) analysis time. The results show that most of the runtime is spent on the NC analysis. For example, in the linear topology with 1,000 flows, 51.3 s are spent on WCD analysis while only 14.9 s are used for RL inference. Similar patterns are observed in the ring (48.1 s vs. 9.7 s) and snowflake (34.2 s vs. 11.7 s) topologies. This observation confirms that NC delay analysis is the main computational bottleneck in the scheduling process, while the RL decision process introduces relatively small overhead.

The experimental results demonstrate that within typical topologies, the compared methods differ in both the number of flows that can be assigned with deterministic NC-based delay guarantees and the computational cost required to obtain such configurations. CBS-DDQN outperforms the baseline

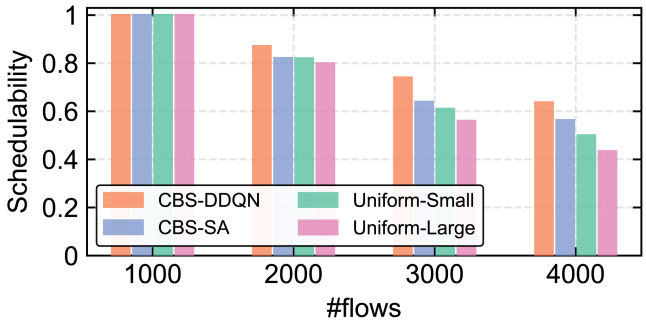


Fig. 8. Scheduling performance in CEV.

scheduling algorithms by efficiently assigning CBS priorities to flows at each hop, thereby improving the satisfiability of E2E delay constraints. This improvement is mainly due to the flexible flow-to-class assignment mechanism, which expands the feasible configuration space and enables the scheduler to find valid configurations that cannot be achieved with fixed class mappings.

C. Effectiveness in Real Network Scenario

The CEV topology represents a realistic large-scale network, which allows us to evaluate the scalability of the proposed method in practical deployment scenarios. Fig. 8 shows the performance of CBS-DDQN on the real-world CEV topology, compared with CBS-SA, Uniform-Small, and Uniform-Large. In this scenario, we focus on schedulability under large-scale traffic conditions. The results show that CBS-DDQN consistently achieves higher schedulability than the baseline methods as the number of flows increases. At 4,000 flows, the schedulability of CBS-DDQN reaches 0.637, which is higher than CBS-SA

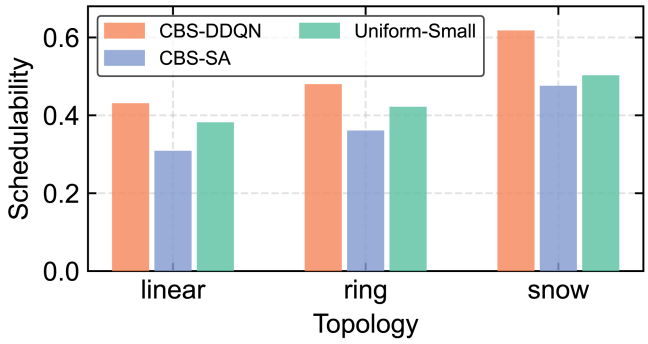


Fig. 9. Schedulability comparison under the deadline–period constraint across different network topologies.

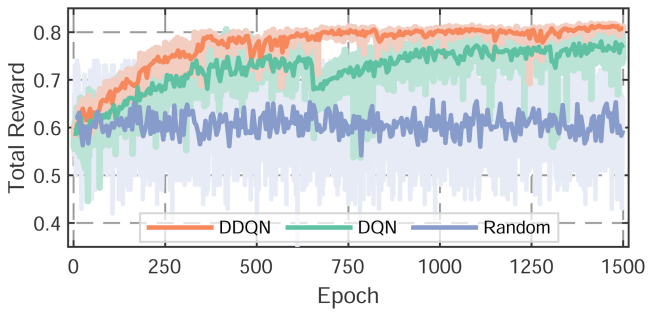


Fig. 10. Convergence performance.

(0.563), Uniform-Small (0.500), and Uniform-Large (0.434). This corresponds to improvements of 7.4%, 13.7%, and 20.3%, respectively. These results highlight that the proposed method can effectively handle large-scale scheduling problems in realistic network topologies by exploring flexible flow-to-class assignments.

D. Experiments Under the Deadline–Period Constraint

In many real-time control systems, the deadline of a task is typically assumed to be no greater than its transmission period. To evaluate the proposed scheduling method under this constraint, we conduct an additional experiment where the deadline of each flow is bounded by its period. The experimental setup follows the configuration described in Section V-A2. The period of each flow is determined by its packet size and transmission rate, and the deadline is randomly generated with its maximum value equal to the period. The evaluation is conducted for the scenario with 1000 flows across typical network topologies. Fig. 9 shows that the proposed CBS-DDQN scheduler achieves higher schedulability than the baseline methods CBS-SA and Uniform-Small, indicating its effectiveness for real-time control scenarios.

E. Convergence Analysis

Fig. 10 compares the reward convergence during training across different CBS priority scheduling methods (DDQN, DQN, and a Random policy). To facilitate comparison, rewards were normalized. DDQN's rewards rise rapidly, stabilizing around 0.8 after 500 epochs. In contrast, DQN converges more slowly, stabilizing between 0.7 and 0.75. The Random method shows greater fluctuations, consistently remaining around 0.6. Overall, DDQN outperforms DQN and Random in convergence speed, reward value, and stability, exhibiting its superior optimization capabilities for CBS priority scheduling.

VI. Conclusion

In this paper, we have addressed the CBS scheduling problem to improve E2E latency satisfiability in IEEE 802.1Qav networks. We have formulated a linear constraint model and proposed `CBS-DDQN`, using a DDQN reinforcement learning approach to optimize CBS parameters across network hops. A fast WCD calculation method was introduced to efficiently compute E2E delay bounds, accelerating model training, and scheduling. Our evaluation results demonstrated `CBS-DDQN`'s improvement of the number of flows meeting latency requirements by 7-28% in large-scale network scenarios, corroborating its effectiveness in managing real-time traffic within TSN environments. Future work includes integrating incremental NC analysis to reduce repeated delay-bound computations and extending the proposed approach to mixed TSN traffic scenarios involving time-triggered traffic, guard-band effects, and frame preemption.

References

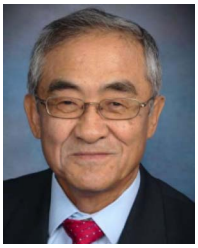
[1] "Time-sensitive networking task group," 2020.[Online]. Available: http://www.ieee802.org/1/pages/tsn.html
[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.
[3] N. Finn, "Introduction to time-sensitive networking," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 22-28, Jun. 2022.
[4] L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1094–1120, Jun. 2019.
[5] D. Bruckner et al., "An introduction to OPC UA TSN for industrial communication systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1121–1131, Jun. 2019.
[6] *Ieee standard for local and metropolitan area networks - virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams*, IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005), pp. C1–72, 2010.
[7] *Ieee standard for local and metropolitan area networks–bridges and bridged networks - amendment 25: Enhancements for scheduled traffic*, IEEE Std 802.1Qbv-2015, pp. 1–57, 2016.
[8] *Ieee standard for local and metropolitan area network–bridges and bridged networks*, IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014), pp. 1–1993, 2018.
[9] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. New York, NY, USA: Springer, 2001.
[10] R. Queck, "Analysis of Ethernet AVB for automotive networks using network calculus," in *Proc. IEEE Int. Conf. Veh. Electron. Saf.*, 2012, pp. 61–67.
[11] J. A. R. De Azua and M. Boyer, "Complete modelling of AVB in network calculus framework," in *Proc. 22nd Int. Conf. Real-Time Netw. Syst.*, 2014, pp. 55–64.
[12] P. Axer, D. Thiele, R. Ernst, and J. Diemer, "Exploiting shaper context to improve performance bounds of Ethernet AVB networks," in *Proc. 51st Annu. Des. Automat. Conf.*, 2014, pp. 1–6.

[13] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers," in *Proc. IEEE Veh. Netw. Conf.*, 2015, pp. 251–258.
[14] D. Maxim and Y.-Q. Song, "Delay analysis of AVB traffic in time-sensitive networks (TSN)," in *Proc. 25th Int. Conf. Real-Time Netw. Syst.*, 2017, pp. 18–27.
[15] L. Zhao, Y. Yan, and X. Zhou, "Minimum bandwidth reservation for CBS in TSN with real-time QoS guarantees," *IEEE Trans. Ind. Informat.*, vol. 20, no. 4, pp. 6187-6198, Apr. 2024.
[16] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "Bandwidth reservation analysis for schedulability of AVB traffic in TSN," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2024, pp. 1–8.
[17] E. Li, F. He, L. Zhao, and X. Zhou, "A SDN-based traffic bandwidth allocation method for time sensitive networking in avionics," in *Proc. IEEE/AIAA 38th Digit. Avionics Syst. Conf.*, 2019, pp. 1–7.
[18] E. Li, F. He, Q. Li, and H. Xiong, "Bandwidth allocation of stream-reservation traffic in TSN," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 741–755, Mar. 2022.
[19] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018.
[20] A. A. Atallah, G. Bany Hamad, and O. Ait Mohamed, "Reliability-aware routing of avb streams in TSN networks," in *Proc. Recent Trends Future Technol. Appl. Intell.: 31st Int. Conf. Ind. Eng. Other Appl. Appl. Intell. Syst.*, 2018, pp. 697–708.
[21] M. Wang, Y. Lu, H. Wang, Z. Chen, and J. Qin, "Load-balanced routing heuristics for bandwidth allocation of AVB flow in TSN," *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 5, pp. 1–19, 2024.
[22] J. Cao, P. J. Cuijpers, R. J. Bril, and J. J. Lukkien, "Independent WCRT analysis for individual priority classes in Ethernet AVB," *Real-Time Syst.*, vol. 54, no. 4, pp. 861–911, 2018.
[23] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of AVB traffic in TSN networks using network calculus," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, 2018, pp. 25–36.
[24] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10, pp. 10291–10302, Oct. 2021.
[25] E. Mohammadpour, E. Stai, and J.-Y. Le Boudec, "Improved credit bounds for the credit-based shaper in time-sensitive networking," *IEEE Netw. Lett.*, vol. 1, no. 3, pp. 136–139, Sep. 2019.
[26] E. Mohammadpour, E. Stai, and J.-Y. Le Boudec, "Improved network-calculus nodal delay-bounds in time-sensitive networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2902–2917, Dec. 2023.
[27] L. Zhao, X. Zhang, H. Feng, and Z. Li, "Incremental performance analysis for accelerating verification of TSN network reconfigurations," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 5, pp. 5475–5490, Oct. 2024.
[28] J. Cao, M. Ashjaei, P. J. Cuijpers, R. J. Bril, and J. J. Lukkien, "An independent yet efficient analysis of bandwidth reservation for credit-based shaping," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst.*, 2018, pp. 1–10.
[29] M. A. Ojewale and P. M. Yomsi, "Routing heuristics for load-balanced transmission in TSN-based networks," *ACM Sigbed Rev.*, vol. 16, no. 4, pp. 20–25, 2020.
[30] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.
[31] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. Hoboken, NJ, USA: Wiley, 2018.
[32] R. L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.
[33] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016, pp. 2094–2100.
[34] T. Schaul, J. Quan, L. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, San Juan, Puerto Rico, May 2–4, 2016, pp. 1–21. [Online]. Available: https://arxiv.org/pdf/1511.05952
[35] Q. Yang et al., "A performance-balanced scheduling algorithm for diverse real-world TSN scenarios," *IEEE Trans. Parallel Distrib. Syst.*, vol. 36, no. 12, pp. 2469–2481, Dec. 2025.
[36] Z3Prover, "Github repository of Z3Prover," 2021. [Online]. Available: https://github.com/Z3Prover/z3
[37] G. Gallo and S. Pallottino, "Shortest path algorithms," *Ann. Operations Res.*, vol. 13, no. 1, pp. 1–79, 1988.
[38] D. Tămaş-Selicean and P. Pop, "Optimization of ttEthernet networks to support best-effort traffic," in *Proc. IEEE Emerg. Technol. Factory Automat.*, 2014, pp. 1–4.
[39] *Iec/ieee Draft International Standard Time-Sensitive Networking Profile for Industrial Automation*, IEC/IEEE P60802/D3.4, May 2025, pp. 1–218, 2025.
[40] *Ieee standard for local and metropolitan area networks–audio video bridging (avb) systems*, IEEE Std 802.1BA-2021 (Revision of IEEE Std 802.1BA-2011), pp. 1–45, 2021.
[41] L. Zhang, G. Sun, R. Liu, W. Quan, H. Yu, and D. Niyato, "Priority-dominated traffic scheduling enabled ATS in time-sensitive networking," *IEEE Trans. Netw. Service Manag.*, vol. 22, no. 1, pp. 470–484, Feb. 2025.
[42] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.
[43] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3th Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7–9, 2015, pp. 1–15. [Online]. Available: https://arxiv.org/pdf/1412.6980

Cheng Long received the B.S. degree from the School of Computer Science & Technology, Beijing Jiaotong University, Beijing, China, in 2023. He is currently working toward the Ph.D. degree with Beijing Jiaotong University. His research interests include time-sensitive networking and real-time embedded systems.

Zonghui Li (Member, IEEE) received the B.S. degree in computer science from the Beijing Information Science and Technology University, Beijing, China, in 2010, and the M.S. and Ph.D. degree from the Institute of Microelectronics and the School of Software, Tsinghua University, Beijing, China, in 2014 and 2019, respectively. He is currently an Associate Professor with the School of Computer Science & Technology, Beijing Jiaotong University, Beijing, China. His research interests include embedded and high performance computing, real-time embedded systems, especially for industrial control networks and fog computing.

Shuxian Jin received the B.S. degree in computer science and technology from Beijing Information Science and Technology University, Beijing, China, in 2022, and the M.S. degree in electronic information from Beijing Jiaotong University, Beijing, China, in 2024. Her research interests focuses on time-sensitive networking.



Kang G. Shin (Life Fellow, IEEE) is currently the Kevin & Nancy O'Connor Professor Emeritus of computer science with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. He has supervised the completion of 93 PhDs, and authored or coauthored about 1,000 technical articles, a textbook and more than 60 patents or invention disclosures. His current research focuses on safe and secure embedded real-time and cyber-physical systems as well as QoS-sensitive computing and networking. He was the recipient of the numerous awards, including 2026 IEEE TC on Distributed Processing (TCDP) Award for Outstanding Technical Achievement, 2023 IEEE TCCPS Technical Achievement Award, 2023 SIGMOBILE Test-of-Time Award, 2019 Caspar Bowden Award for Outstanding Research in Privacy Enhancing Technologies, and Best Paper Awards from 2023 VehicleSec, 2011 ACM International Conference on Mobile Computing and Networking, 2011 IEEE International Conference on Autonomic Computing, 2010 and 2000 USENIX Annual Technical Conferences, as well as 2003 IEEE Communications Society William R. Bennett Prize Paper Award and the 1987 Outstanding IEEE Transactions of Automatic Control Paper Award, Research Excellence Award, in 1989, Outstanding Achievement Award in 1999, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award, in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty), Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002, 2003 IEEE RTC Technical Achievement Award, and 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He has chaired the Michigan Computer Science and Engineering Division for 4 years starting 1991, and also several major conferences, including 2009 ACM MobiCom, and 2005 ACM/USENIX MobiSys. He was a co-founder of a couple of startups, licensed some of his technologies to industry, and was an Executive Advisor for Samsung Research.

Daquan Zhang received the B.S. degree in artificial intelligence from Beijing Jiaotong University, Beijing, China, in 2023, whre he is currently working toward the M.S. degree in computer science and technology. His research interests include ray tracing and time-sensitive networking.

Zixiao Wang received the B.S. degree in computer science and technology from the College of Information Science and Engineering, Hohai University, Nanjing, China, in 2023. He is currently working toward the M.S. degree with Beijing Jiaotong University, Beijing, China. His research interests include industrial real-time networks, 5G and traffic scheduling.

Zhibo Pang (Senior Member, IEEE) received the M.B.A. degree in innovation and growth from the University of Turku, Turku, Finland, in 2012 and the Ph.D. degree in electronic and computer systems from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2013. He was an Adjunct Professor with the University of Sydney. He is currently with the School of Advanced Manufacturing and Robotics, Peking University, Beijing, China. He works on embodied intelligence, robotics, control, computing, communication, and electronics for Industry 4.0 and Healthcare 4.0. He has many productized research results and 23 granted patents in USA, Europe, or Japan. He is a member of IEEE IES Industry Activities Committee, a Steering Committee Member of the IEEE IoT Technical Community, the Chair of IEEE Technical Committee on Cloud and Wireless Systems for Industrial Applications, and the Co-Chair of the IEEE TC on Industrial Informatics. He was the recipient of the "Inventor of the Year Award" by ABB Corporate Research Sweden, three times in 2016, 2018, and 2021. He is an Associate Editor of IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN INDUSTRIAL ELECTRONICS, and *IEEE Internet of Things Magazine*.