Burst-Aware Time-Triggered Flow Scheduling With Enhanced Multi-CQF in Time-Sensitive Networks

Dong Yang, *Member, IEEE*, Zongrong Cheng, *Student Member, IEEE*, Weiting Zhang, *Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*— Deterministic transmission guarantee in time-sensitive networks (TSN) relies on queue models (such as CQF, TAS, ATS) and resource scheduling algorithms. Thanks to its ease of use, the CQF queue model has been widely adopted. However, the existing resource scheduling algorithms of CQF model only focus on periodic time-triggered (TT) flows without consideration of bursting flows. Considering that the bursting flows often carry high-priority data in real systems, in this paper we investigate the mixed-flow (i.e., TT and bursting flows) scheduling problem in CQF-based TSN aiming to maximize the number of schedulable flows and system load balance while satisfying the deterministic demands of delay, jitter, and reliability for both TT and bursting flows. Unfortunately, it is challenging to schedule the mixed flows with the original CQF model because of the huge difference between TT and bursting flows. To resolve this problem, we firstly design an enhanced Multi-CQF model to satisfy the basic demands of bursting flows sent at any time without affecting the deterministic transmission of TT flows. Given the complexity of mixed-flow scheduling and the proposed queue model, it is difficult for traditional algorithms to fully utilize network resources. Thus, we further propose a time-correlated DRL resource scheduling (TimeDRS) algorithm to optimize the resource allocation. TimeDRS can be extended to other time-related resource scheduling scenarios, such as TDMA-based scheduling. Experimental results demonstrate that our proposed approaches can greatly reduce frame loss and end-to-end latency for bursting flows, and well balance runtime and schedulability compared with state-of-the-art benchmarks.**

***Index Terms*— Time-sensitive networks, resource scheduling, enhanced Multi-CQF, deep reinforcement learning.**

I. Introduction

RECENTLY, there is a significant development of technologies for time-sensitive networks (TSN), which is proposed by IEEE 802.1 TSN task group. Evolving from traditional Ethernets, TSN targets at providing the industrial applications, such as vehicular, avionic, and automotive control systems, with deterministic bounded end-to-end latency, jitter, and high reliability [1], [2], [3].

The deterministic transmission guarantee of TSN is based on network resource scheduling in the unit of data flow [4]. According to whether the data flow information can be known in advance, the data flow can be classified into time-triggered flows and bursting flows. Both types of flows are capable of carrying high-priority data [5]. For example, based on standard of International Society of Automation [6], the time-triggered (TT) flows carry real-time periodic monitoring or controlling data, and the bursting flows carry unexpected controlling or safety data that has a higher requirement for low-latency [7]. Hence, an available and practicable TSN system must support mixed-flow (i.e., TT and bursting flows) scheduling.

TSN standards define several queue models for network resource scheduling, such as cyclic queuing and forwarding (CQF) model, time-aware shaper (TAS), and asynchronous traffic shaper (ATS). Among these queue models, the CQF model has been widely adopted thanks to its ease of use [8]. To fully utilize the network resources, various TT flow scheduling algorithms have been proposed for CQF system, such as satisfiability modulo theories (SMT) [9], Greedy-3q [10], and injection time planning (Tabu-ITP) [11]. However, owing to the complexity of mixed-flow scheduling, none of the related works have considered jointly scheduling both the TT and bursting flows by CQF model. Inspired by this, we investigate the mixed-flow (i.e., TT and bursting flows) scheduling problem in CQF-based TSN aiming to maximize the number of schedulable flows and load balance of the whole system while satisfying the deterministic transmission demands of end-to-end latency, jitter, and high reliability for both TT and bursting flows.

In essence, the original CQF model is designed to support deterministic transmission of TT flows by preserving enough network resources. However, the bursting flows cannot be well supported by the original CQF model. This is because the sending time of bursting flows is uncertain, and they may arrive later than the expected receiving duration, which causes frame loss. One way to improve the reliability of bursting flows is to extend the expected receiving duration of original CQF model [12], but the end-to-end latency increases significantly for both bursting and TT flows. Therefore, a major conflict arises between the reliability and low-latency. To resolve this conflict and support a collaborative transmission for the mixed

Manuscript received 5 June 2022; revised 27 February 2023; accepted 26 March 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Iosifidis. Date of publication 13 April 2023; date of current version 19 December 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB2901302, in part by the National Natural Science Foundation of China under Grant 62201029, and in part by the China Post-Doctoral Science Foundation under Grant 2022M710007 and Grant BX20220029. *(Corresponding author: Weiting Zhang.)*

Dong Yang, Zongrong Cheng, Weiting Zhang, and Hongke Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: dyang@bjtu.edu.cn; zrcheng@bjtu.edu.cn; wtzhang@bjtu.edu.cn; hkzhang@bjtu.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, N2L 0B5, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TNET.2023.3264583

flows, we design an enhanced Multi-CQF model. By properly selecting the queuing offset on each hop, the high reliability and low latency demands of bursting flows can be satisfied without affecting deterministic transmission of TT flows.

Due to the complexity of mixed-flow scheduling and our proposed enhanced Multi-CQF model, traditional scheduling algorithms, such as SMT and heuristics, often take too much time and effort to compute the scheduling results. To improve the efficacy of mixed-flow scheduling, we design a time-correlated deep reinforcement learning (DRL) resource scheduling (TimeDRS) algorithm. Different from traditional DRL-based resource scheduling algorithms that separately schedule time interval and queuing resources [13], [14], TimeDRS embeds with a two-dimensional time-resource matrix to efficiently learn the relationship between different time intervals and queues, then jointly allocates time interval and queuing resources for each flow. TimeDRS can also be extended to more generic time-related network resource scheduling scenarios, such as joint time-queue scheduling for TSN using queue models other than CQF, joint time-frequency domain scheduling in wireless networks, or other types of time division multiple access (TDMA) based resource scheduling in wired and wireless networks.

Compared with the original CQF, our enhanced Multi-CQF model can reduce the frame loss rate from 13.45% to 0.97% under heavy loads of bursting flows. As a comparison, if we want to achieve the same frame loss rate (i.e., 0.97%) by extending the receiving duration of original CQF model, the end-to-end latency of bursting flows will increase by 2.18x than our enhanced Multi-CQF model. Furthermore, by introducing TimeDRS algorithm to our enhanced Multi-CQF model, the number of schedulable TT flows can be increased by 2x, 60%, 77.9%, and 35% than SMT [9], DeepCQF [13], Greedy-3q [10], and Tabu-ITP [11] algorithms, respectively. A good trade-off between the runtime and schedulability can also be achieved. The contributions of this paper are as follows:

- **Enhanced Multi-CQF model.** We design an enhanced Multi-CQF model to support the deterministic transmission for mixed flows. By assigning the queuing offset for each flow, this model can guarantee the high reliability and low end-to-end latency of bursting flows without affecting TT flows. The deterministic guarantee of enhanced Multi-CQF model is further validated on our TSN testbed in real environment. As far as we know, this is the first queue model in CQF-based system taking the bursting flows into account.
- **Time-correlated DRL resource scheduling algorithm.** We propose a time-correlated DRL resource scheduling algorithm, namely TimeDRS, to allocate available network resources. Embedded with a time-resource matrix module, the relationship between time intervals and network resources can be efficiently learned, and the overall scheduling performance can be improved. TimeDRS can be used in generic time-related resource scheduling scenarios, such as TDMA-based resource scheduling.
- **Open-source simulator.** We build up a publicly accessible CQF-based TSN simulator as well as several benchmark algorithms to flexibly support the scheduling environment with different number of cyclic queues.[1] Furthermore, we evaluate the performance of the proposed queue model on the simulator.

The remainder of this paper is organized as follows. In Section II, the backgrounds of original CQF model are presented. Section III establishes an enhanced Multi-CQF model for TSN to support the transmission for mixed flows. Section IV and V further design a time-correlated DRL resource scheduling algorithm, TimeDRS, to improve the scheduling QoS of entire system. Section VI evaluates the performance of our proposed mechanisms. The related works, followed by the conclusions are presented in Section VII and Section VIII.

II. Background and Challenge

This section introduces the principles and challenges of mixed-flow scheduling in the original CQF model.

A. Background of CQF Model

Generally, the physical topology of a TSN network can be abstracted as an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. As described by Eq. (1), $\mathcal{V}$ is a set of vertices including end stations $\mathcal{H}$ and switches $\mathcal{S}$ in the network, and $\mathcal{E}$ refers to all the connected links between any two vertices v_i and v_j. Each link $e_i \in \mathcal{E}$ supports the full-duplex data transmission,

$$\begin{aligned}\mathcal{E} &= \{[v_i,v_j]|\forall v_i,v_j \in \mathcal{V}, v_i \neq v_j\}, \mathcal{V} = \mathcal{H} \cup \mathcal{S},\\ \mathcal{G}(\mathcal{V},\mathcal{E}) &= \{v_i,[v_k,v_j]|\forall v_i \in \mathcal{V}, \forall [v_k,v_j] \in \mathcal{E}\},\\ &\quad i,j,k = 1,2,\ldots,\mathcal{N}. \end{aligned} \tag{1}$$

The periodic TT flows $f_i \in \mathcal{F}$ in network $\mathcal{G}(\mathcal{V},\mathcal{E})$ are featured by ID, period, source and destination hosts, frame size, predefined path, frame number, requirements of delay and jitter, and sending offset, described by

$$\begin{aligned} f_i = \{&id, period, src, dst, size, path, \mho_i, r_d, r_j, offset\},\\ &\forall f_i \in \mathcal{F}, i = 1,2,\cdots,|\mathcal{F}|. \end{aligned} \tag{2}$$

On the other hand, the mismanaged TT flows are inevitable in real environment because of the unexpected flow sending time [14]. In addition, the safety-critical flows need higher priority to prevent harmful behaviors in the system triggered by events without periodicity. These two types of flows belong to the bursting flows f_i^- in this paper. The features of bursting flows f_i^- are described by

$$\begin{aligned} f_i^- = \{&id, src, dst, size, path, \mho_i, r_d, r_j, offset\},\\ &\forall f_i^- \in \mathcal{F}, i = 1,2,\cdots,|\mathcal{F}|. \end{aligned} \tag{3}$$

Deterministic flow transmission by CQF relies on queue operations. The original CQF model maintains two queues, denoted as Q_1 and Q_2 respectively, cyclically performing the en-queue and de-queue operations by controlling the gates

[1]The CQF-based TSN simulator is publicly accessible for related researchers at https://github.com/zcheng19/CQF-based_TSN

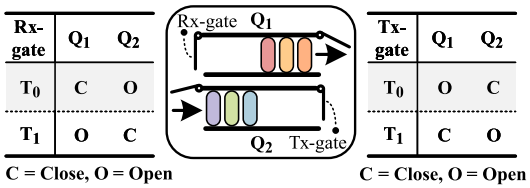


Fig. 1. Queuing discipline for the original 2-buffer CQF model. The cyclic queuing and forwarding mechanism is implemented by repeatedly changing the open and closed status for Rx-gate and Tx-gate attached to each queue.

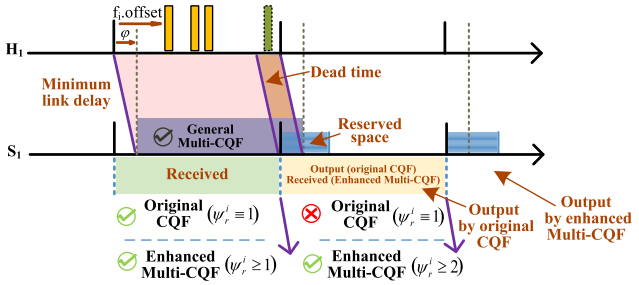


Fig. 2. Comparison of different variants for CQF model. The flows are sent from host H_1 to the adjacent switch S_1.

attached to each queue. The time intervals within a hyper period are the cycle time for queue operations [8]. Fig. 1 assumes that two time intervals T_0, T_1 are in a hyper period. During even interval T_0, the Q_1 opens Tx-gate to send the flows, and Q_2 opens Rx-gate to receive the flows. In odd interval T_1, the execution operates conversely. Thus, the queue operations for different time intervals can be described by

$$\begin{cases} Q_1 = 0, Q_2 = 1; \ \ if \ \ T_0, \\ Q_1 = 1, Q_2 = 0; \ \ if \ \ T_1, \end{cases} \tag{4}$$

where 0 denotes the de-queue operation, otherwise 1. To provide the deterministic transmission latency, jitter, and high reliability, all the sending flows from the upstream nodes are required to be received by the downstream nodes within one time interval [8], and continued to be forwarded during the next time interval. Let $delay_i^{[v_s,v_d]}$ be the real end-to-end latency, and $jitter_i^{[v_s,v_d]}$ be the jitter (i.e., latency variance) of the system. Then the end-to-end latency and jitter for CQF model can be bounded with

$$\begin{cases} (\mathcal{N}-1) \times \mathcal{T} \leq delay_i^{[v_s,v_d]} \leq (\mathcal{N}+1) \times \mathcal{T}, \\ 0 \leq jitter_i^{[v_s,v_d]} \leq 2 \times \mathcal{T}, \end{cases} \tag{5}$$

where $\mathcal{N}$ is the number of hops along the flow path $f_i.path$, and $\mathcal{T}$ is the size of time interval. To guarantee the deterministic flow transmission performance, CQF establishes a common time reference shared by all the TSN entities based on a precise network-wide time synchronization mechanism, which is supported by IEEE 802.1AS [15] stand-alone standard with the profile of generic precision time protocol (gPTP). The core idea of gPTP is to adjust the clock offset between the Grand Master (GM) and each slave node to ensure that the slave nodes can work well with GM at the same pace.

B. Challenges for Mixed-Flow Scheduling in CQF Model

The mixed-flow scheduling in the original CQF model mainly faces the following three challenges.

One major challenge is to ensure the high reliability for both bursting and TT flows. On the one hand, the flow transmission principle in the original CQF model cannot avoid frame loss for bursting flows. As shown in Fig. 2, the original CQF model requires that all the flows sent from host H_1 must be received by switch S_1 within one time interval. However, impacted by flow transmission delay and unexpected sending time, the bursting flows may be sent during the dead time and cannot be accommodated by only one time interval. This leads to frame loss. Dead time is caused by the bursting flows arriving later than the closing time of the Rx-gate attached to the receiving queue. On the other hand, it is difficult to guarantee the reliability of mixed flows as the flows may exceed the capacity of the time interval when they gather into the same queue. Therefore, how to avoid the frame loss for mixed-flows in CQF model is a critical problem to be investigated.

Another great challenge is to guarantee the end-to-end latency of mixed flows with different priorities. The bursting flows have the highest latency priority compared with other types of flows. As a result, any interference on bursting flows that increases end-to-end latency is expected to be avoided. Besides, different latency demands of TT flows should be satisfied by the queue model. However, the original CQF model only provides two cyclic queues without distinguishing TT and bursting flows that have different latency demands. Although the existing Multi-CQF models introduce multiple queues, and solve the packet loss problem of TT flows transmitted through the long-link DetNet [10], they still do not consider any bursting flows or different latency demands of TT flows. Therefore, a new queuing and forwarding strategy of the queue model should be designed.

Finally, given the complexity of mixed-flow scheduling and queue model, traditional scheduling algorithms, such as SMT and heuristics, are very difficult to fully utilize the available network resources [16]. As a result, the QoS of the whole system, including the number of schedulable flows and load balance are degraded. Compared with the traditional algorithms, the DRL algorithm is regarded as a promising technology to intelligently adapt to the dynamic environment, because it can iteratively learn the good policies from experience by trial and error [17]. However, the investigations of DRL-based scheduling methods for the mixed-flow transmission in TSN is still at the early stage, and need a further exploration. We believe that solving the challenges mentioned above will bring a leap for TSN in the aspects of practicability and intelligence.

III. Enhanced Multi-CQF Model: Providing a Guarantee of High Reliability and Latency

In this section, we propose an enhanced Multi-CQF model, which supports both bursting and TT flows. The proposed mixed-flow scheduling theorems provide the analysis of flow scheduling characteristics in the enhanced Multi-CQF model.

A. Design of Enhanced Multi-CQF Model

To enhance the original CQF, we establish $\mathcal{K}$ ($\mathcal{K} \geq 3$) queues on each switch port cyclically performing the en-queue and de-queue operations, namely enhanced Multi-CQF model. During each time interval, only one queue has the privilege to send the flows, denoted as $Q_i = 0, 0 \leq i \leq \mathcal{K}-1$, and the rest of queues only receive the flows, denoted as $Q_j = 1, j \neq i, j = 0, 1, \ldots, \mathcal{K}-1$. This queuing principle effectively avoids any conflicts of flows sent from different queues. Specifically, in each time interval T_ξ, the operations on different queues are defined by

$$\begin{cases} Q_{\xi\%\mathcal{K}} = 0, \xi \geq 0, \xi\%\mathcal{K} \leq \mathcal{K}-1, \\ Q_j = 1, j \neq \xi\%\mathcal{K}, j = 0, 1, \ldots, \mathcal{K}-1, \end{cases} \tag{6}$$

and the queue operations can be achieved by controlling the Rx-gate and Tx-gate attached to each queue based on time synchronization. For example, in time interval T_0, the Q_0 opens Tx-gate to send the flows, and the rest of queues open Rx-gate to receive the flows.

With the queuing principle of Eq. (6), the flow receiving strategy is further defined to support the mixed-flow transmission in the enhanced Multi-CQF model. We introduce the queuing offset to describe the flow receiving strategy.

Definition 1 (Queuing Offset): The queuing offset tag ψ_j on the queue Q_j refers to the distance from Q_j to the queue $Q_{\xi\%\mathcal{K}}$ that is used for sending flows during the current time interval T_ξ. The queuing offset of $Q_{\xi\%\mathcal{K}}$ is denoted as $\psi_{base} = 0$, while for the rest of queues, the queuing offset is sequentially accumulated by 1 in a circulating shift manner.

Based on above definition, the queuing offset ψ_j for each queue Q_j in time interval $\mathcal{T}_\xi$ can be described by

$$\begin{cases} \psi_j = \psi_{base} + (j - \xi\%\mathcal{K} + \mathcal{K}); \ \ if \ \ j - \xi\%\mathcal{K} < 0, \\ \psi_j = \psi_{base} + (j - \xi\%\mathcal{K}); \ \ if \ \ j - \xi\%\mathcal{K} \geq 0, \\ j = 0, 1, \ldots, \mathcal{K}-1, \xi \geq 0. \end{cases} \tag{7}$$

To ensure the high reliability, any bursting flows sent during the dead time $\mathcal{T}_d$ from the upstream nodes can choose the receiving queue of the downstream nodes tagged with $\psi_r \geq 2$ to prevent the frame loss as shown in Fig. 2, such that

$$\psi_r^i \times \mathcal{T} \geq f_i^-.offset + delay_i^{[v_x, v_y]} + \delta, \tag{8}$$

where ψ_r^i is the queuing offset chosen by flow f_i^-, and $\mathcal{T} < f_i^-.offset + \mathcal{T}_d$. The Eq. (8) extends the expected receiving duration of the downstream nodes from $\mathcal{T}$ to $\psi_r^i \times \mathcal{T}$, and any bursting flow f_i^- transmitted in dead time $\mathcal{T}_d$ can be accommodated. As such, the flow receiving strategy can be set up in real CQF-based TSN system as follows. Because the bursting flows are generally triggered by the hosts, they have to choose $\psi_r^i \geq 2$ of the first hop to avoid the frame loss. On next several hops, the stored bursting flows from the upstream nodes can be immediately sent when the Tx-gate opens, which means they are transmitted before dead time $\mathcal{T}_d$ with high reliability guarantee. For this case, the bursting flows will select $\psi_r^i = 1$ of the downstream nodes to reduce the transmission latency between any two hops. This is an effective way to achieve the "quick-forwarding" without waiting for several time intervals. For the TT flows, we aim to avoid the interference to bursting flows, and flexibly adjust the forwarding speed at each hop based on different latency demands of TT flows. Therefore, TT flows can select $\psi_r^i \geq 1$ at each hop based on the traffic volume and latency demands.

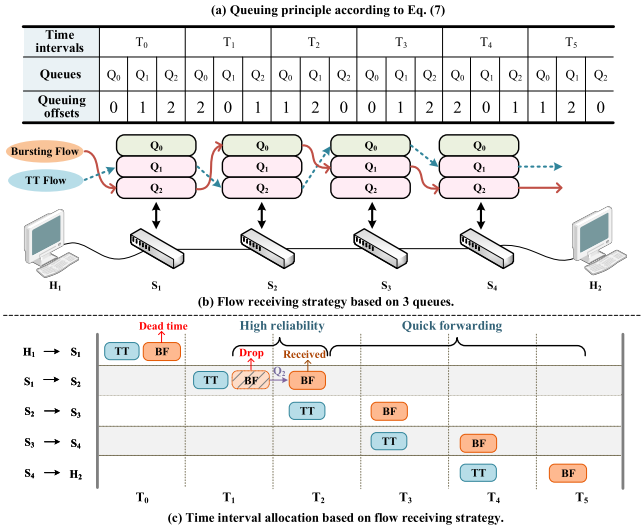


Fig. 3. An example for the enhanced Multi-CQF model.

Fig. 3 gives an example to describe the enhanced Multi-CQF model. In time interval T_0, host H_1 sends the bursting and TT flows to H_2 passing by 4 TSN switches. We set $\mathcal{K} = 3$ for the queue model. To ensure the high reliability, the bursting flow selects $\psi_r^i = 2$ during T_0, i.e., Q_2 according to Eq. (7), to receive the frames at first hop S_1, such that Eq. (8) is satisfied. After two time intervals, in T_2, the bursting flow is forwarded by S_1, and then assigned $\psi_r^i = 1$ of next 3 hops to reduce the transmission latency between any two adjacent hops. The TT flow selects $\psi_r^i = 1$ at each hop to meet its deterministic end-to-end latency, jitter, and high reliability demand without interfering bursting flows. If more time-delay can be tolerated, TT flow can also selects $\psi_r^i > 1$ to leave more spaces for other flows that need low-latency guarantee.

B. Mixed-Flow Scheduling Theorems

Based on our enhanced Multi-CQF model, several mixed-flow scheduling theorems are further presented as follows.

Theorem 1: The end-to-end latency and jitter for any flow $\forall f_i \in \mathcal{F}$ transmitted from v_s to v_d is bounded with

$$\begin{cases} delay_i^{[v_s, v_d]} \leq \sum_{n=1}^{\mathcal{N}+1} \psi_r^{i(n)} \times \mathcal{T}, \\ delay_i^{[v_s, v_d]} > \left(\sum_{n=1}^{\mathcal{N}} \psi_r^{i(n)} - 1 \right) \times \mathcal{T}, \\ jitter_i^{[v_s, v_d]} < \left(\psi_r^{i(\mathcal{N}+1)} + 1 \right) \times \mathcal{T}, \end{cases} \tag{9}$$

in our system model, where $\mathcal{N}$ denotes the number of hops traversed by flow f_i along the path. Based on this notation, all the nodes along the flow path are numbered as $0, 1, \ldots, \mathcal{N}+1$ from the source to destination host. $\psi_r^{i(n)}$ is the queuing offset that the flow f_i chooses on node n.

Proof: For $\forall f_i \in \mathcal{F}$ transmitted through the link $[v_x, v_y] \subseteq [v_s, v_d]$, where v_y is the next hop of v_x, the

constraint

$$f_i.offset + delay_i^{[v_x,v_y]} + \delta \le \psi_r^{i(n)} \times \mathcal{T}, \\ \Rightarrow delay_i^{[v_x,v_y]} \le \psi_r^{i(n)} \times \mathcal{T} - f_i.offset - \delta, \quad (10)$$

should be satisfied, where $f_i.offset$ is the clock offset between the Tx-gate opening time on node v_x and the actual sending time of flow f_i, and it has $0 \le f_i.offset \le \mathcal{T}$. $\delta \approx 0$ holds for the precise time synchronization. Therefore, the maximal transmission latency $delay_i^{[v_x,v_y]}.max = \psi_r^{i(n)} \times \mathcal{T}$ holds when $f_i.offset = 0$, deduced by Eq. (10). Then, $delay_i^{[v_s,v_d]}$ has the upper limit, described as

$$delay_i^{[v_s,v_d]} \le \sum_{[v_x,v_y]=[v_s,v_{s+1}]}^{[v_{d-1},v_d]} delay_i^{[v_x,v_y]}.max \\ = \sum_{n=1}^{\mathcal{N}+1} \psi_r^{i(n)} \times \mathcal{T}. \quad (11)$$

Taking the forwarding process into consideration. Because f_i sent from the source node v_s is stored into the queue tagged with $\psi_r^{i(1)}$ of the first hop v_{s+1}, and it cannot be forwarded until the Tx-gate opens, there should exist

$$f_i.offset + delay_i^{[v_s,v_{s+1}]} + \delta = \psi_r^{i(1)} \times \mathcal{T}, \\ \Rightarrow delay_i^{[v_s,v_{s+1}]} = \psi_r^{i(1)} \times \mathcal{T} - f_i.offset - \delta, \quad (12)$$

which means the queuing delay will be additionally appended to $delay_i^{[v_s,v_{s+1}]}$ if f_i arrives before $Q_r = 0$. When $f_i.offset = \mathcal{T}$, $delay_i^{[v_s,v_{s+1}]}.min = (\psi_r^{i(1)} - 1) \times \mathcal{T}$ holds. On the rest of links $[v_x, v_y]$, where $[v_x, v_y] \ne [v_{d-1}, v_d]$, the $delay_i^{[v_x,v_y]}$ of f_i should be at least $\psi_r^{i(n)} \times \mathcal{T}$ according to the queuing principle defined by Eq. (6). $delay_i^{[v_{d-1},v_d]} = \rho > 0$ holds as the propagation delay exists. Therefore, the lower bound of end-to-end latency $delay_i^{[v_s,v_d]}$ can be described by

$$delay_i^{[v_s,v_d]} \ge delay_i^{[v_s,v_{s+1}]}.min + \sum_{n=2}^{\mathcal{N}} \psi_r^{i(n)} \times \mathcal{T} + \rho \\ = \left(\sum_{n=1}^{\mathcal{N}} \psi_r^{i(n)} - 1\right) \times \mathcal{T} + \rho \\ > \left(\sum_{n=1}^{\mathcal{N}} \psi_r^{i(n)} - 1\right) \times \mathcal{T}. \quad (13)$$

Thus, the jitter can be computed by subtracting Eq.(11) and Eq.(13), which is $jitter_i^{[v_s,v_d]} < \left(\psi_r^{i(\mathcal{N}+1)} + 1\right) \times \mathcal{T}$. □

Theorem 2: For the dead time $\mathcal{T}_d$ existing in any time interval T_ξ of the original CQF model, a necessary and sufficient condition to eliminate $\mathcal{T}_d$ by our enhanced Multi-CQF model is to choose the queuing offset ψ_r^i for receiving flows that satisfies

$$\psi_r^i \ge \frac{delay_i^{[v_x,v_y]}}{\mathcal{T}} + 1, \quad (14)$$

such that for $\forall T_\xi$, it has $\mathcal{T}_d = 0$. Where, $\forall f_i \in \mathcal{F}$, and node v_y is the next hop of node v_x.

Proof: We first prove that $\forall T_\xi$, $\mathcal{T}_d \ne 0$ holds in the original CQF model. Based on the method of counter-proof, it can be assumed that $\exists T_k$, it has $\mathcal{T}_d = 0$ in CQF. That means for any flow f_i transmitted in time interval T_k, there must be

$$\begin{cases} f_i.offset + delay_i^{[v_x,v_y]} + \delta \le \mathcal{T}, \\ delay_i^{[v_x,v_y]} > 0, \delta \approx 0. \end{cases} \quad (15)$$

Actually, for the last sending flow f_l in $\mathcal{T}_k$, who has $f_l.offset = \mathcal{T}$, the transmission latency is

$$\Theta_l = f_l.offset + delay_l^{[v_x,v_y]} + \delta \\ = \mathcal{T} + delay_l^{[v_x,v_y]} + \delta \\ > \mathcal{T}. \quad (16)$$

However, the Eq. (16) contradicts with Eq. (15), which proves the existence of $\mathcal{T}_d$ in any time interval T_ξ. Then, when the Eq. (14) holds, we have

$$\mathcal{T} + delay_i^{[v_x,v_y]} + \delta \le \psi_r^i \times \mathcal{T}, \quad (17)$$

where Eq. (17) is the identical deformation of Eq. (14). For any flow with $f_i.offset \le \mathcal{T}$, it has $f_i.offset + delay_i^{[v_x,v_y]} + \delta \le \mathcal{T} + delay_i^{[v_x,v_y]} + \delta \le \psi_r^i \times \mathcal{T}$. That means all the flows transmitted in T_ξ can be received without any frame loss, that is $\mathcal{T}_d = 0$. In turn, with the elimination of $\mathcal{T}_d$, $\forall f_i$ transmitted in T_ξ, where $f_i.offset \le \mathcal{T}$, can be accommodated by the receiving duration with the length of $\psi_r^i \times \mathcal{T}$, that is $f_i.offset + delay_i^{[v_x,v_y]} + \delta \le \psi_r^i \times \mathcal{T}$. As a result, in case of $f_i.offset = \mathcal{T}$, $\mathcal{T} + delay_i^{[v_x,v_y]} + \delta \le \psi_r^i \times \mathcal{T}$ should be satisfied. Thus, the Eq. (14) holds. The sufficiency and necessity of Eq. (14) are proved. □

After the dead time has been eliminated by choosing the proper queuing offset in Thm. 2, the flow aggregation that affects the network performance should be further avoided, analyzed in Thm. 3 and Thm. 4.

Definition 2: Flow aggregation. The flow aggregation of any two flows $f_i, f_j \in \mathcal{F}$ means there exists frames of f_i and f_j gathered into the same sending queue, denoted as $f_i \succ f_j$.

Theorem 3: For any two periodic TT flows $f_i, f_j \in \mathcal{F}$ mapped with the same queuing offset, that is $\psi_r^i = \psi_r^j$, $f_i \succ f_j$ will happen on node v_k when existing integers $m \ge 0, n \ge 0$, such that the Eq. (18) and Eq. (19) hold,

$$\left|(f_i.offset + m \times f_i.period + delay_{i_m}^{[v_s^i,v_k]}) - (f_j.offset \right. \\ \left. + n \times f_j.period + delay_{j_n}^{[v_s^j,v_k]})\right| \le \psi_r^i \times \mathcal{T}, \quad (18)$$

$$\left|(f_i.offset + m \times f_i.period + delay_{i_m}^{[v_s^i,v_k]} + \Delta_{i_m}^{[v_k,v_p]}) \right. \\ \left. - (f_j.offset + n \times f_j.period + delay_{j_n}^{[v_s^j,v_k]} + \Delta_{j_n}^{[v_k,v_p]})\right| \\ \le \mathcal{T}, \quad (19)$$

where $delay_{i_m}^{[v_s^i,v_k]}$ means the delay between source node v_s^i and switch node v_k for transmitting the m-th frame of flow f_i along the path $f_i.path$. $\Delta_{i_m}^{[v_k,v_p]}$ refers to the queuing delay for the m-th frame of f_i at the port of node v_k, whose adjacent node is v_p.

Proof: Upon $f_i \succ f_j$ at node v_k, there exists frames arriving at the same port of v_k during the same receiving time interval. Thus, Eq. (18) holds. Besides, after queuing, the variation of sending offsets between f_i and f_j on the NIC

should be no more than $\mathcal{T}$ before the Tx-gate closes, and Eq. (19) holds. In turn, if only Eq. (18) holds, meaning that f_i and f_j are received by v_k during the same interval $t \in [0, \psi_r^i \times \mathcal{T}]$, it still cannot guarantee that the flow f_i and f_j aggregate into the same queue. For example, we let the m-th frame of f_i send within T_ξ from the upstream node v_{k-1}, and it arrives to v_k during time interval $T_{\xi+1}$. In this case, it will be mapped to queue $Q_{\mathscr{I}(\xi,\psi_r^i)}$, where $\mathscr{I}(\cdot)$ is the index mapping function,

$$\mathscr{I}(\xi,\psi_r^i) = \begin{cases} \xi\%\mathcal{K} + \psi_r^i; & if \ \ \psi_r^i \le \mathcal{K} - \xi\%\mathcal{K} - 1, \\ \xi\%\mathcal{K} - \mathcal{K} + \psi_r^i; & if \ \ \psi_r^i > \mathcal{K} - \xi\%\mathcal{K} - 1, \end{cases} \tag{20}$$

we then let the n-th frame of f_j send during $T_{\xi+1}$ from v'_{k-1}, and it arrives at v_k during the same time interval $T_{\xi+1}$, such that it satisfies with Eq. (18) equivalent to

$$\left|(f_i^m.offset_{v_{k-1}} + delay_{i_m}^{[v_{k-1},v_k]}) - (f_j^n.offset_{v'_{k-1}} + \mathcal{T} + delay_{j_n}^{[v'_{k-1},v_k]})\right| \le \psi_r^i \times \mathcal{T}, \tag{21}$$

and the frame f_j^n will be mapped into queue $Q_{\mathscr{I}(\xi+1,\psi_r^j)}$. As $\psi_r^i = \psi_r^j$, we have $Q_{\mathscr{I}(\xi,\psi_r^i)} \neq Q_{\mathscr{I}(\xi+1,\psi_r^j)}$, which means they cannot be aggregated into the same queue. When Eq. (19) holds, the difference of sending offsets between f_i^m and f_j^n can be limited within $\mathcal{T}$, and they are sent during the same time interval from v_k. Therefore, $f_i \succ f_j$ holds. □

The scheduling performance of TT flows is mainly affected by the flow aggregation described by Thm. 3. In particular, if an existing queue contains too much data, the frames are more likely to be dropped because the time interval has finite capacity to accommodate flows. In our queue model, the queuing offset ψ_r^i for each flow is determined mainly by their different levels of end-to-end latency demands according to Thm. 1. If we dynamically change ψ_r^i to avoid flow aggregation, it greatly increases the latency variance and even dissatisfies the end-to-end latency demands. Thus, scheduling time interval resources without affecting ψ_r^i would be a better way to to avoid the flow aggregation, described by Thm. 4.

Theorem 4: For the m-th frame of f_i and n-th frame of f_j that has $f_i^m \succ f_j^n$, a sufficient condition for avoiding the aggregation is to postpone the sending offset of flow f_i by $\tilde{o}$ from source node v_s^i, such that the Eq. (22) and Eq. (23) hold,

$$\left|(f_i.offset + \tilde{o} + m \times f_i.period + delay_{i_m}^{[v_s^i,v_k]}) - (f_j.offset + n \times f_j.period + delay_{j_n}^{[v_s^j,v_k]})\right| > \psi_r^i \times \mathcal{T}, \tag{22}$$

$$f_i.offset + \tilde{o} \le f_i.period, \tag{23}$$

where $m, n \ge 0$, $m, n \in Z$, $\psi_r^i = \psi_r^j$.

Proof: As analyzed by Thm. 3, if $f_i^m \succ f_j^n$ appears, Eq. (18) must hold. On the contrary, if Eq. (22) holds, which means the difference of arriving time between f_i^m and f_j^n on v_k exceed $\psi_r^i \times \mathcal{T}$, the two frames cannot be stored into the same queue Q_j as the Rx-gate of Q_j will close before f_i^m comes. As they are stored into different queues, the sending offset on the link $[v_k, v_p]$ cannot satisfy with Eq.(19), either. Thus, the aggregation of f_i^m and f_j^n is avoided. Moreover, the sending offset of f_i cannot be postponed for more than a period of f_i to avoid the conflict with other frame f_i^e generated in other period. Thus, Eq. (23) holds as another condition. □

As analyzed by Thm. 4, an effective way to avoid the flow aggregation is to postpone the sending time of f_i from source host. Based on our enhanced Multi-CQF model, such an operation can be transformed into allocating the time intervals for each flow f_i from source hosts, and the scheduling performance highly depends on the quality of time interval resource allocation policy. The complexity of allocating time interval resources is analyzed by Thm. 5.

Theorem 5: The optimization problem $\mathbf{P}_0$ based on the time interval allocation policy for each flow f_i described by Eq. (26) is NP-hard.

Proof: First, the time interval allocation optimization problem $\mathbf{P}_0$ is not NP because when the solution is given, it is uncertain whether the given solution is optimal in a polynomial time. Second, if the number of time intervals is set to 1 as a special case, the problem can then be converted to a choice of which flow should be mapped to that time interval, such that

$$Maximize \ \sum_{i=1}^{n} r_i \times x_i, \tag{24}$$

and it is subject to

$$\begin{cases} \sum_{i=1}^{n} f_i.size \times x_i \le T_\xi.c, \\ x_i \in \{0,1\}, n \le |\mathcal{F}|, \end{cases} \tag{25}$$

where x_i indicates whether flow f_i is mapped into the time interval, r_i is the reward for mapping f_i into this time interval, and $T_\xi.c$ refers to the capacity of T_ξ. Eq. (24) and Eq. (25) are equivalent to 0-1 knapsack problem that has been proved as an NP-hard. Therefore, the 0-1 knapsack problem can be reduced to our problem, meaning that the time interval allocation optimization problem is also the NP-hard. □

IV. Problem Formulation and Transformation

In this section, we formulate the optimization problem as the management of time interval and queuing resources in the enhanced Multi-CQF model, and then the formulated problem is transformed into a Markov Decision Process (MDP).

A. Problem Formulation

We set up a multi-objective optimization problem $\mathbf{P}_0$ for mixed-flow scheduling in the enhanced Multi-CQF model. The objective goals of $\mathbf{P}_0$ include maximizing the schedulable number of TT flows and system load balance. These goals can be achieved by properly allocating the sending time intervals for each TT flow to avoid the flow aggregation, according to Thm. 3 and Thm. 4. Moreover, the end-to-end latency of bursting flows is also expected to reach the minimum, where the "quick-forwarding" queues should be assigned to bursting flows referred to Thm. 1 and Thm. 2. Let $R_\xi(\cdot)$ denote the value of scheduling each flow, and $\Phi_\xi(f_i)$ indicates whether f_i is allocated into T_ξ. As such, the objective goals in problem $\mathbf{P}_0$ are formulated by the cumulative values.

$$\mathbf{P}_0 : Maximize \ \frac{1}{|\mathcal{F}|} \sum_{i=1}^{|\mathcal{F}|} \sum_{\xi=0}^{\lambda-1} \mathcal{R}_\xi(f_i, \lambda, \mathcal{F}) \times \Phi_\xi(f_i), \tag{26}$$

where $\Phi_\xi(f_i) \in \{0,1\}$. If flow f_i is allocated into T_ξ, then $\Phi_\xi(f_i) = 1$, otherwise, $\Phi_\xi(f_i) = 0$. The value $\mathcal{R}_\xi(f_i, \lambda, \mathcal{F})$ depends on the number of scheduled flows computed by $\mathscr{A}(\cdot)$, load balance of $\mathscr{C}(\cdot)$, and the real end-to-end latency of bursting flows $delay_{i^-}^{[v_s,v_d]}$. Hence,

$$\mathcal{R}_\xi(f_i,\lambda,\mathcal{F}) = \alpha \times \mathscr{A}(f_i,\lambda) - \beta \times \mathscr{C}(\lambda,\mathcal{F}) + o \times \frac{1}{delay_{i^-}^{[v_s,v_d]}}, \tag{27}$$

where $0 < \alpha, \beta, o < 1$ are the discounting factors. If all the frames of flow f_i can be successfully planned, $\mathscr{A}(f_i,\lambda)$ will be 1, otherwise, it should be 0. Hence, it has

$$\mathscr{A}(f_i,\lambda) = \begin{cases} 1; & if \quad \sum_{j=1}^{\mho_i}\sum_{t=0}^{\lambda-1}\Phi_t(f_i^j) = \mho_i, \\ 0; & if \quad otherwise. \end{cases} \tag{28}$$

where $\mho_i$ denotes the frame number of f_i appeared in a hyper period. In particular, it is required that one frame can only be mapped into one corresponding time interval, that is

$$\sum_{t=0}^{\lambda-1}\Phi_t(f_i^j) \le 1, \Phi_\xi(f_i^j) \ge 0, \Phi_\xi(f_i^j) \in \mathcal{Z}, \tag{29}$$

$\mathscr{C}(\lambda,\mathcal{F})$ measures the performance of load balance by computing the standard deviation of time interval utilization,

$$\mathscr{C}(\lambda,\mathcal{F}) = \sqrt{\frac{1}{\lambda}\sum_{t=0}^{\lambda-1}(\chi_t - \bar{\chi})^2}, \tag{30}$$

where χ_t denotes the resource usage of T_t, that is

$$\chi_t = \frac{\sum_{m=1}^{|\mathcal{F}|}\sum_{s=1}^{\mho_m}\Phi_t(f_m^s)\times f_m.size}{T_t.c}, \tag{31}$$

and $\bar{\chi} = \frac{1}{\lambda}\sum_{t=0}^{\lambda-1}\chi_t$. Here, $delay_{i^-}^{[v_s,v_d]}$ is the real end-to-end latency of the bursting flows. The shorter the delay is, the better the value $R_\xi(\cdot)$ can acquire. Besides, the problem $\mathbf{P}_0$ is subject to several network resource constraints, including hyper period, time interval, and bandwidth capacity. The hyper period defined in TSN is the least common multiple (LCM) of all flow periods [13] ensuring that the flows can be periodically scheduled. Hence, the hyper period is constrained by

$$C_1 : \mathcal{P} = LCM(\mathcal{F}.periods). \tag{32}$$

As each time interval is the granularity for scheduling in our enhanced Multi-CQF model, all the flow periods should be divisible by the predefined time interval size, described by condition C_2. Thus, the maximal size of each time interval should be the greatest common divisor (GCD) of flow periods, constrained by

$$C_2 : \mathcal{P}\%\mathcal{T} = 0, f_i.period\%\mathcal{T} = 0, \tag{33}$$

$$C_3 : \mathcal{T} \le GCD(\mathcal{F}.periods). \tag{34}$$

To guarantee the high reliability of transmitting the bursting flows, the receiving duration $\psi_r^i \times \mathcal{T}$ of the cyclic queue with ψ_r^i should be large enough to accommodate all the flows, i.e., TT and bursting flows, sent from the upstream nodes. Considering the worst case that the bursting flows sent at the end of time interval, $\psi_r^i \times \mathcal{T}$ must be larger than the summation of a whole time interval $\mathcal{T}$, transmission delay, other propagation delays $\mathcal{D}$, and the time synchronization bias δ. Hence, the time interval constraint is described by

$$C_4 : \psi_r^i \times \mathcal{T} \ge \mathcal{T} + \frac{max(\mathcal{F}.sizes)}{\mathcal{B}} + \mathcal{D} + \delta, \tag{35}$$

where the transmission delay depends on the bandwidth and the maximal frame size in the network, which is $\frac{max(\mathcal{F}.sizes)}{\mathcal{B}}$. It is required that the total size of sending flows during each time interval cannot exceed its maximal capacity $T_\xi.c$. Thus, the time interval capacity constraint is

$$C_5 : 0 \le \sum_{m=1}^{|\mathcal{F}|}\sum_{s=1}^{\mho_m} f_m.size \times \Phi_\xi(f_m^s) \le T_\xi.c, \tag{36}$$

and the flow transmission constraints are described by C_6-C_{10}. In constraint C_6, it requires that the sending time of each TT flow should be smaller than its flow period, which can protect the frames in different periods against preempting the limited resources of NIC at the same time.

$$C_6 : 0 \le f_i.offset \le f_i.period. \tag{37}$$

To guarantee deterministic transmission latency, the worst end-to-end delay of each flow should be no later than the latency demand. Therefore, the latency constraint is

$$C_7 : \sum_{n=1}^{\mathcal{N}+1}\psi_r^{i(n)} \times \mathcal{T} \le f_i.r_d. \tag{38}$$

The jitter of each flow is the latency variance of all its frames. We let $f_i^j.\mathcal{L}_{v_s}^{v_d}$ be the real end-to-end latency of the j-th frame in flow f_i. It is required that the gap between end-to-end latency of each frame $f_i^j.\mathcal{L}_{v_s}^{v_d}$ and the average flow latency does not exceed the required jitter. Thus, the jitter constraint is described by

$$C_8 : \left| f_i^j.\mathcal{L}_{v_s}^{v_d} - \frac{1}{\mho_i}\sum_{s=1}^{\mho_i} f_i^s.\mathcal{L}_{v_s}^{v_d} \right| \le f_i.r_j. \tag{39}$$

The high reliability of each flow is required by C_9, where the loss of frames cannot be larger than the target threshold. The mapping function $\Phi_t(f_i^s)$ is used to compute the number of successfully scheduled frames. More variable constraints of C_1-C_9 are described in C_{10}, and other application related constraints can be flexibly appended by users.

$$C_9 : \left| \sum_{s=1}^{\mho_i}\sum_{t=0}^{\lambda-1}\Phi_t(f_i^s) - \mho_i \right| \le 1 \times 10^{-6}, \tag{40}$$

$$C_{10} : \forall i \in \{1,2,\ldots,|\mathcal{F}|\}, \forall j \in \{1,2,\ldots,\mho_i\}, \lambda = \frac{\mathcal{P}}{\mathcal{T}}, \forall \xi \in \{0,1,\ldots,\lambda-1\}. \tag{41}$$

B. Problem Transformation

To solve the problem $\mathbf{P}_0$ in the different scheduling scenarios, we transform $\mathbf{P}_0$ into an MDP [18]. With a fully centralized architecture proposed by IEEE 802.1Qcc [19], the controller in the control plane can be considered as an agent.

By collecting the flow information and network status at each step, the agent learns the optimal time interval allocation policies from the environment. The core elements of MDP, including the state, action, and reward are designed as follows.

State: The time interval utilization of network and flow features are covered in the observation space $\mathcal{O}$. In particular, at step t, the state $o_t = o_t^{util} \cup o_t^{flow}$, where $o_t \in \mathcal{O}$. o_t^{util} is comprised by a set of occupied time interval capacity on each queue, which is described by

$$\begin{aligned} o_t^{util} = \{ & \mathscr{H}_{T_0}^{Q_{n_0}^{p_0}}, \mathscr{H}_{T_0}^{Q_{n_1}^{p_1}}, \ldots, \mathscr{H}_{T_0}^{Q_{n_{J-1}}^{p_{K-1}}}, \mathscr{H}_{T_0}^{Q_{n_J}^{p_K}}, \\ & \mathscr{H}_{T_1}^{Q_{n_0}^{p_0}}, \mathscr{H}_{T_1}^{Q_{n_1}^{p_1}}, \ldots, \mathscr{H}_{T_1}^{Q_{n_{J-1}}^{p_{K-1}}}, \mathscr{H}_{T_1}^{Q_{n_J}^{p_K}}, \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ & \mathscr{H}_{T_L}^{Q_{n_0}^{p_0}}, \mathscr{H}_{T_L}^{Q_{n_1}^{p_1}}, \ldots, \mathscr{H}_{T_L}^{Q_{n_{J-1}}^{p_{K-1}}}, \mathscr{H}_{T_L}^{Q_{n_J}^{p_K}} \}, \end{aligned} \tag{42}$$

where $\mathscr{H}_{T_\ell}^{Q_{n_j}^{p_k}}$ symbolizes the resource usage of available space within each time interval T_ℓ when storing the flows into each receiving queue on port p_k at node n_j. o_t^{flow} includes the features of f_t that arrives at step t, described by

$$\begin{aligned} o_t^{flow} = \{ & f_t.id, f_t.period, f_t.src, f_t.dst, f_t.path, \\ & f_t.r_d, f_t.r_j, f_t.\mathfrak{V}_t, f_t.size \}. \end{aligned} \tag{43}$$

Action: The action space $\mathcal{A}$ consists of λ sending time intervals waiting to be planned in a hyper period $\mathcal{P}$. We let a_{f_t} denote one of the sending time intervals in $\mathcal{A}$ for the flow f_t to insert, which means flow f_t will send the data from time interval a_{f_t}. Because each a_{f_t} under the observation o_t has its corresponding Q-value $Q(o_t, a_{f_t})$, at each step t, the agent will choose the final action a_t among all the available a_{f_t} according to the mapping policy $\pi : \mathcal{O} \rightarrow \mathcal{A}$ [20], and the mapping policy is described by

$$a_t = \begin{cases} \arg\max\limits_{a_{f_t} \in \mathcal{A}} Q(o_t, a_{f_t}); \quad if \quad rand \geq \epsilon, \\ a_{f_t}; \quad otherwise, \\ \epsilon \in [0, 1], \end{cases} \tag{44}$$

where $a_t \in \mathcal{A}$ denotes allocating T_t for f_t to make the Q-value $Q(o_t, a_{f_t})$ optimal in the probability of ϵ. And $Q(o_t, a_{f_t})$ is computed by the action-value function $Q^\pi(o, a)$ that reflects the long-term revenue by interacting with the environment, formatted as Bellman equation as follows [21], [22].

$$Q^\pi(o, a) = E\left[r_{t+1} + \gamma Q^\pi(o_{t+1}, a_{f_{t+1}}) | o_t = o, a_{f_t} = a, \pi \right], \tag{45}$$

where the immediate reward denoted as r_{t+1} can be obtained by the agent after a_t has been taken.

Reward: The immediate shaping reward r_{t+1} can be received when the agent takes action a_t, and it acts as a feedback to conduct the agent performing better at the next step $t+1$. The design of r_{t+1} is relevant to the objective function [23], where r_{t+1} is the summation of r_{t+1}^a, denoting the capability of flow scheduling, and r_{t+1}^b, referring to the difference of load balance between any two consecutive time steps. Hence, the equation of r_{t+1} can be described as

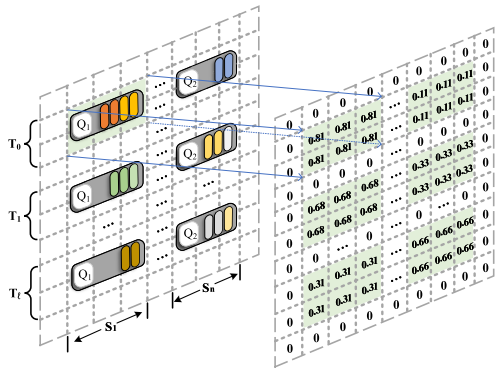


Fig. 4. The state array of the time interval resource utilization.

$$\begin{cases} r_{t+1} = \mu^a \times r_{t+1}^a + \mu^b \times r_{t+1}^b, \\ r_{t+1}^a = \left| \Gamma_{t+1} - |\mathcal{F}| \right| - \gamma^a \times \left| \Gamma_t - |\mathcal{F}| \right|, \\ r_{t+1}^b = \left| o_{t+1}^{util}.max - \Xi \right| - \gamma^b \times \left| o_t^{util}.max - \Xi \right|, \\ r_{t+1}^a \leq 0, r_{t+1}^b \geq 0, \end{cases} \tag{46}$$

where $\mu^a, \mu^b \in [-1, 0]$, $\gamma^a, \gamma^b \in [0, 1]$ are the parameters for fine-tuning. Γ_t is the number of scheduled flows at step t. $o_t^{util}.max$ denotes the maximum usage of time interval resources, and Ξ is the ideal occupancy of the time interval. Besides, when the task is done, the additional reward is given based on the scheduling performance. With the core elements of MDP designed above, the DRL algorithm can then be used to solve the problem $\mathbf{P}_0$.

V. TimeDRS: A Time-Correlated DRL Resource Scheduling Algorithm

In this section, the TimeDRS is proposed as a DRL-based algorithm to solve the mixed-flow scheduling problem both from offline and online in the enhanced Multi-CQF model. Several tricks are designed to further improve the schedulability. Without loss of generality, TimeDRS can be extended to many other time-related network resource scheduling scenarios, such as joint time-queue scheduling for TSN using queue models other than CQF, joint time-frequency domain scheduling in wireless networks, or other types of TDMA-based resource scheduling in wired and wireless networks.

A. Designing Tricks

Given the discrete problem $\mathbf{P}_0$, the TimeDRS uses a value-based framework. Several tricks are designed to help the agent capture key features and train better policies. Specifically, given that time interval and queuing resources are correlated in the enhanced Multi-CQF model, one-dimensional fully-connected neural network cannot well learn the relationship between time intervals and queues because the time interval and queuing resources are separately input into the neural network. Thus, a two-dimensional time-resource matrix is set up in Trick 1 to jointly consider the time intervals and queues.

Trick 1: *Create two-dimensional time-resource matrix.* This matrix is composed by both queuing and time interval

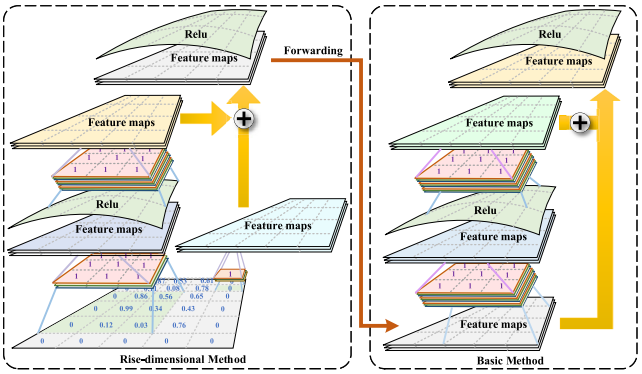


Fig. 5. Designed extraction method of feature maps through residual blocks.

resources of the network. Each row represents a time interval, and each column denotes a cyclic queue of node. This matrix is set to the first channel of the state for the agent's observation. When a new flow comes, the resource utilization of each queue within various time intervals will be changed, and the state can be transited. By introducing other time-related network resources, such as frequency and computing resources, into each column of the matrix, TimeDRS can be extended to more generic time-related network resource scheduling scenarios, such as joint time-queue scheduling for TSN using queue models other than CQF, joint time-frequency domain scheduling in wireless networks, or other types of TDMA-based resource scheduling in wired and wireless networks. To further enhance the key features, we vectorize each element $\mathscr{H}_{T_\ell}^{Q_{n_j}^{p_k}}$ to a $u \times m$ array $\nu_{q_k}^{(\ell)}$, which maps the usage of each time interval into each queue q_k. Zeros padding is made between any two queues, such that a $p \times p$ state matrix $\mathscr{P}(o_t^{util})$ is re-created, depicted by Fig. 4.

To capture the key features of the two-dimensional state, the convolution neural network is used because it can be more adapted to the two-dimensional input. Moreover, utilizing the residual neural network (ResNet) as the enhanced structure of convolutional neural network can well solve the problem of gradient disappearance or explosion in the process of back propagation. As a result, ResNet provides a more stable training process, and quick convergence performance. Hence, the Trick 2 is introduced.

Trick 2: *Utilize the modern residual neural network.* The ResNet consists of a several residual blocks shown in Fig. 5. Two methods are designed for residual blocks. The rise-dimensional method introduces 1×1 kernels to increase the number of channels for state array, and the basic method operates same convolution from observed state. The extracted information passes from rise-dimensional block to the basic block. Based on ResNet, a general mapping strategy from x_ℓ to the deeper layer x_L can be expressed by

$$x_L = h(x_\ell) + \sum_{i=\ell}^{L-1} \mathscr{F}(x_i; \mathcal{W}_i), \tag{47}$$

where $\mathscr{F}(\cdot)$ is the end-to-end mapping function including convolution, batch normalization, and activation. $h(\cdot)$ denotes 1×1 convolution operation used to increase the dimension of x_ℓ. Thus, The back propagation (BP) function can be deduced.

$$\begin{aligned}\frac{\partial l_o}{\partial x_\ell} &= \frac{\partial l_o}{\partial x_L}\frac{\partial x_L}{\partial x_\ell} = \frac{\partial l_o}{\partial x_L}\left[\frac{\partial h(x_\ell)}{\partial x_\ell} + \frac{\partial}{\partial x_\ell}\sum_{i=\ell}^{L-1}\mathscr{F}(x_i;\mathcal{W}_i)\right] \\ &= \frac{\partial l_o}{\partial x_L}\frac{\partial h(x_\ell)}{\partial x_\ell} + \frac{\partial l_o}{\partial x_L}\frac{\partial}{\partial x_\ell}\sum_{i=\ell}^{L-1}\mathscr{F}(x_i;\mathcal{W}_i),\end{aligned} \tag{48}$$

where l_o is the loss function, and $\frac{\partial l_o}{\partial x_L}\frac{\partial h(x_\ell)}{\partial x_\ell}$ enhances the BP of gradient by skipping several intermediate layers.

Algorithm 1 TimeDRS for Scheduling TT Flows Based on the Enhanced Multi-CQF Model

Input: Empty replay buffer $\mathcal{D}$, Initial network parameters ω and ω^-, Target network update frequency $\mathscr{L}$, Replay buffer size $\mathcal{N}_r$, Batch size $\mathcal{D}_b$, Action space $\mathcal{A}$.

Output: Optimal policy $\pi^* : \mathcal{O} \rightarrow \mathcal{A}$.

for *episode* $e \in \{1, 2, \ldots, \mathcal{M}\}$ **do**
  Reserve queuing and time interval resources for bursting flows by **Algorithm 2**;
  Deduce ψ_r^i for each TT flow by Thm. 1 based on their different latency demands;
  Initialize and wrap the state array o;
  for *step* $t \in \{1, 2, \ldots, \mathcal{U}\}$ **do**
    Select a_t from $Q(o_t, a_{f_t}; \omega)$ according to $\epsilon - greedy$ strategy and expert guidance;
    Allocating flow f_t into sending time interval a_t;
    Store flow f_i into $Q_{\mathscr{I}(\xi,\psi_r^i)}$ at each T_ξ;
    Transit o_t to the next state o_{t+1};
    Obtain an immediate reward r_{t+1};
    Store tuple $(o_t, a_t, r_{t+1}, o_{t+1}, done)$ into $\mathcal{D}$;
    Replace the old tuples if $|\mathcal{D}| > \mathcal{N}_r$;
    Sample a batch of $\mathcal{D}_b$ tuples $(o, a, r, o', done)$ from replay buffer $\mathcal{D}$;
    Define $\zeta_{max}(o'; \omega) = \arg\max_{a' \in \mathcal{A}} Q(o', a'; \omega)$;
    Determine $y_j = \begin{cases} r, \ if \ done, \\ r + \gamma^c Q'(o', \zeta_{max}(o'; \omega); \omega^-), \ otherwise; \end{cases}$
    Perform a gradient decent step with loss $l_o = E[(y_j - Q(o, a; \omega))^2]$;
    Reset $\omega^- = \tau\omega + (1 - \tau)\omega^-$ every $\mathscr{L}$ steps;
    if *done* **then**
      break;
    end
  end
end

In the training process, there exists many invalid actions based on the constraints of problem $\mathbf{P}_0$. To assist the agent in distinguishing the valid or invalid actions under different observations, the action array marked with legitimacy for each action should be merged into the observed state, which is described by Trick 3.

Trick 3: *Design collaborative working mechanism between the state and action.* To help the agent actively choose the valid actions based on the constraints of problem $\mathbf{P}_0$,

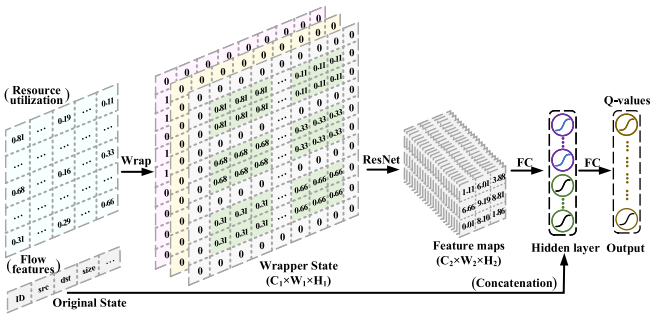


Fig. 6. Decision process based on the input state from environment.

we concatenate $\mathscr{P}(o_t^{util})$ in Fig. 4 with action array in the dimension of channels to collaboratively update the state and action at each step. The action array includes two channels, one denotes the queuing status during each time interval T_ξ based on Eq. (6), and the other marks the valid sending time intervals for flow f_t constraint by Eq. (37). The valid actions are encoded by 1, while those invalid ones are filled with 0. As a result, $\mathscr{P}(o_t^{util})$ is wrapped by three channels.

To facilitate the training process, we introduce an expert to predict the state transition before the selected action is taken. It can efficiently guide the agent to choose a better action at each step, described by Trick 4.

Trick 4: *Expert guidance.* The expert can predict the operation condition of the system before the agent selects an action. If the selected action will degrade the mixed-flow scheduling performance, the expert can guide the agent to take a better action. This mechanism can effectively assist the agent learning high-quality policies.

B. Working Mechanism

The principle of our proposed TimeDRS algorithm can be summarized by the following five parts.

Initialization: In line 1-4 of Alg. 1, at each episode, we assign the queuing offset for each bursting flow based on high reliability and low end-to-end latency requirements, and reserve enough time interval capacity to send the bursting data. In particular, the ψ_r^i on the first hop should be constrained by $\psi_r^i \geq 2$ so that any bursting flows generated from the hosts can be successfully received by the queue, described by Eq. (8). In the subsequent forwarding process, the queued bursting flows from the upstream node will select $\psi_r^i = 1$ queue of the downstream node for receiving. This policy not only ensures the high reliability, but also reduces the end-to-end latency for bursting flows. The TT flows choose the ψ_r^i of receiving queue mainly based on their different end-to-end latency demands according to Thm. 1. The global network resources observed by Eq. (42) and Eq. (43) are also initialized.

State Transition: The state transition is illustrated in line 5-10 of Alg. 1. At each step t, the agent captures the state information o_t, and uses the evaluation ResNet to abstract the feature maps from the wrapper state $\mathscr{P}(o_t^{util})$. After going through the fully-connected layer, the other part of state o_t^{flow} is concatenated, as shown in Fig. 6. Then based on Q-values output by the neural network, the agent decides which time interval T_t should be allocated for the TT flow f_t. After the flow is transmitted through each hop along the path and stored by the corresponding queues, o_t transits to the next state o_{t+1}, and the reward r_{t+1} described by Eq. (46) can be acquired to assist the agent learning better policies.

Algorithm 2 Dealing With Bursting Flows

Input: Time interval set $\mathscr{T}$, bursting flows set $\mathcal{F}_b$.
Output: Rest space of each $T_i \in \mathscr{T}$.
1 **for** *each interval* $T_i \in \mathscr{T}$ **do**
2 $\quad space_i \leftarrow alloc\_space(T_i)$;
3 $\quad list_{rest}.push((T_i.c - space_i))$;
4 **end**
5 **for** *each flow* $f_i^- \in \mathcal{F}_b$ **do**
6 $\quad \psi_r^{i^-} \leftarrow fast\_queue(f_i^-)$;
7 $\quad Q_{\mathscr{I}(\xi,\psi_r^{i^-})} \leftarrow assign(f_i^-, \psi_r^{i^-}, T_\xi)$ based on the time interval T_ξ and allocated $\psi_r^{i^-}$;
8 **end**
9 return $list_{rest}$;

Exploring approach: To achieve the balance of exploration and exploitation, TimeDRS applies the ϵ-greedy strategy in choosing a_t according to Eq. (44), where ϵ decreases in a regular pattern with episodes e_p, which is described by

$$\epsilon = I_0 - \frac{|I_0 - I_e|}{\mathcal{D}_c} \times e_p, \tag{49}$$

where $I_0, I_e \in (0, 1]$ represent the upper and lower bounds of ϵ, respectively, and $I_0 > I_e$. $\mathcal{D}_c$ is a large positive number and it is constrained by $\forall e_p \leq \mathcal{D}_c$.

Sampling management: As described in line 11-13 of Alg. 1, TimeDRS utilizes the off-policy updating method. Thus, the replay buffer is introduced to sample the transition trajectories $\{o_t, a_t, r_{t+1}, o_{t+1}, done\}$ obtained by the agent at each step. Using the off-policy strategy can not only stabilize the training process, but also improve the sampling efficiency, because the batch-sampling based training method covers a larger state space for observation.

Training process: To overcome the instability caused by the self excitation effect of bootstrap in Bellman function, we introduce the target network with the same structure of the evaluation ResNet. Starting from line 14 in Alg. 1, the agent samples a batch of trajectories and minimizes the loss function l_o by updating network parameters ω, illustrated by

$$l_o = \frac{1}{\mathcal{D}_b} \sum_{j=1}^{\mathcal{D}_b} (y_j - Q(o_j, a_j; \omega))^2, \tag{50}$$

which is a Q-value based updating approach. And y_j is the target value calculated by

$$y_j = \begin{cases} r_j; \ \ if \ done, \\ r_j + \gamma^c Q'(o'_j, \arg\max_{a'_j \in \mathcal{A}} Q(o'_j, a'_j; \omega); \omega^-); \ otherwise. \end{cases} \tag{51}$$

Then the parameters of target network ω^- can be updated every $\mathscr{L}$ steps by computing the moving average value

between evaluation and target parameters, that is $\omega^- = \tau\omega + (1-\tau)\omega^-$, where $\tau \in [0,1]$ is the temperature to smooth the process of update [24].

VI. Implementation and Analysis

In the different scheduling scenarios, the changes for algorithms, time interval sizes, flow period sets, and bandwidth of bursting flows can affect the network performance. In this section, we evaluate the mixed-flow scheduling performance on our CQF-based TSN simulator with Python. For the simulated environment, we assume that the deterministic transmission can be guaranteed with the enhanced Multi-CQF model. To further validate the assumption of determinism in real environment, the end-to-end latency and jitter are measured by transmitting different flows on our TSN testbed.

A. Performance Evaluation in Simulated Environments

Settings for TSN network: The Orion Crew Exploration Vehicle (CEV) network [25] is selected as the benchmark topology in our experiments, which is combined with star, tree, and ring topologies, connecting with 31 end stations and 13 switches, depicted in Fig. 7, where the dotted lines are added links to leave some redundancy during the scheduling. We assume that all the switches enable the function of enhanced Multi-CQF. 1,000 TT flows are scheduled under different network parameter settings, and the bursting flows are randomly generated in each time interval. The detailed network parameters are listed in Table I.

Settings for TimeDRS: TimeDRS uses two ResNet neural networks with the same structure for evaluating and updating, respectively. Based on our experience, we set 8 residual blocks $B_1 \sim B_8$ for each ResNet to deal with the wrapper state, whose parameters are set by Table III. The wrapper state is set to $3 \times 44 \times 44$ transformed from the original state. The extracted information is forwarded from B_1 to B_8, where B_1, B_3, B_5 and B_7 use the rise-dimensional method to abstract key features, while the rest of blocks are basic ones used to enhance the abstracted information. For the training process, we set 10,000 episodes, each includes 1,000 steps of iteration. The other parameters of TimeDRS are described in Table I.

Settings for SOTA algorithms: The SOTA scheduling algorithms, such as DeepCQF, SMT, Greedy-3q, and Tabu-ITP, are compared to analyze the scheduling performance.[2] DeepCQF uses two fully-connected hidden layers, each with 1,000 neurons, and the other learning parameters are also described in Table I. The SMT returns the satisfiable results of problem computed by Z3-solver tool for Python within an acceptable time set by 30 minutes. And Greedy-3q always chooses the sending time interval with the maximal capacity along the path. The exchanging mode is used in Tabu-ITP, where the size of tabu list is set to 500, and the iterations along with repetitions are 10,000 and 200, respectively. During each iteration, no more than 5 flows can be shifted between the flow sets to generate the candidates.

[2]We reproduce SOTA algorithms, including DeepCQF, SMT, Greedy-3q, and Tabu-ITP, described in their corresponding papers. The scheduling are extended from the original CQF to the enhanced Multi-CQF model.

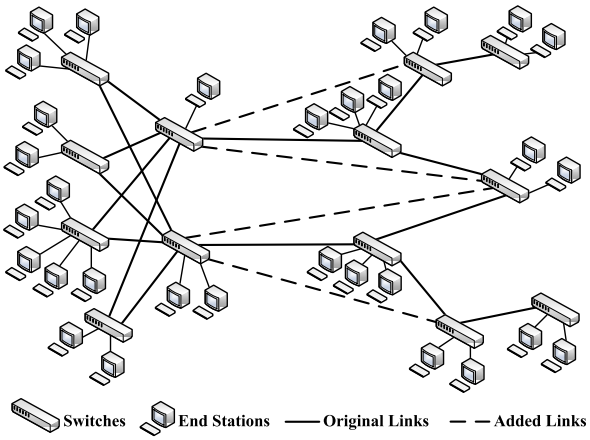


Fig. 7. Topology of the Orion CEV network.

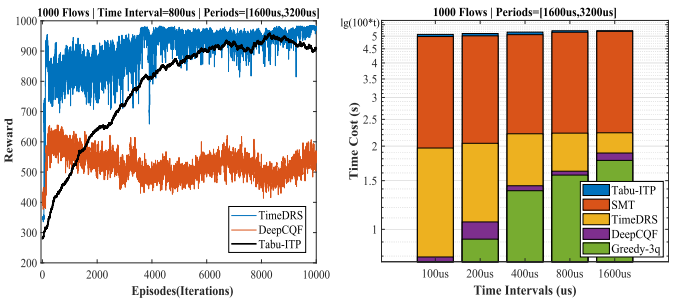


Fig. 8. Experimental results. (a) Training or searching process of different algorithms; and (b) Computational time cost comparison.

The experiments are conducted in the off-line and on-line scheduling scenarios with our enhanced Multi-CQF model, whose results and data analysis are described as follows.

Convergence comparison: The iterative searching process is shown in Fig. 8(a), where the time interval $\mathcal{T}$ is 800us, and the flow periods are {1600us, 3200us}. Compared with Tabu-ITP, the convergence speed of TimeDRS is relatively faster because it can better extract the key features from the regular experience. Furthermore, TimeDRS hits the highest reward after converging. On the contrary, the performance of DeepCQF does not meet our expectations, which is because the fully-connected layer in DeepCQF only observes one-dimensional input (i.e., queuing resources) at each step, and it cannot find the relationship between time intervals and queues.

Time cost comparison: The computational time cost of various algorithms with different time interval sizes is compared in Fig. 8(b). TimeDRS, DeepCQF, and Greedy-3q can schedule the flows in a few seconds, while Tabu-ITP and SMT need dozens of minutes to compute the scheduling results. For better visualization, we take the logarithm function to deal with the computational time spent by each algorithm. Specifically, TimeDRS spends more time computing the ResNet than Greedy-3q and DeepCQF. Fortunately, as the time interval size gets larger, the computational time cost of TimeDRS tends to be closer to Greedy-3q and DeepCQF because Greedy-3q and DeepCQF require more time to make inference for the increased flows.

TABLE I
SIMULATION PARAMETERS

Network Para.	Val.			Network Para.	Val.		Learning Para.	Val.	Learning Para.	Val.
$\mathcal{T}$	100us	200us	400us	$\mathcal{F}.sizes$	50B∼1KB		$\mathcal{D}.length$	1e5	Adam.l_r	1e-5
	800us	1600us		$\mathcal{P}$	1600us	3200us	$\mathcal{D}_b$	32	$\mathscr{L}$	50
$\mathcal{F}.periods$	[800us,1600us,3200us]			ψ_r^i	2		I_0	1	μ^a	-1
	[1600us,3200us]	[800us,1600us]		Host Number	31		I_e	0.1	μ^b	-0.1
$\mathcal{B}$	1,000Mbps			Switch Number	13		$\mathcal{D}_c$	1e4	γ^a,γ^b	1
Cyclic Queues	3			Reservation	50Kb/T_ξ		τ	0.95	γ^c	0.99

TABLE II
SIMULATION PARAMETER DESCRIPTIONS

Para.	Meaning	Para.	Meaning
$\mathcal{T}$	time interval size	D_c	Decay of ϵ
$\mathcal{F}.periods$	Flow periods	τ	Training temperature
$\mathcal{B}$	Link Bandwidth	Adam.lr	Learning rate
$F.sizes$	Frame size	$\mathscr{L}$	Update interval
$\mathcal{P}$	Hyper period	μ^a	Scheduling weight
ψ_r^i	Queuing offset	μ^b	Load balance weight
$\mathcal{D}.length$	Buffer size	γ^a	Flow number factor
$\mathcal{D}_b$	Batch size	γ^b	Reward factor
I_0	Upper limit of ϵ	γ^c	Discounting factor
I_e	Lower limit of ϵ	B_n	Residual blocks

TABLE III
THE PARAMETERS OF RESIDUAL BLOCKS

(Rise dimension)	B_1	B_3	B_5	B_7
Conv1	Kernel size=3×3; Padding=1; Stride=2			
Conv2	Kernel size=3×3; Padding=1; Stride=1			
Conv3	Kernel size=1×1; Padding=0; Stride=2			
Output size	64×22×22	128×11×11	256×6×6	512×3×3
(Basic method)	B_2	B_4	B_6	B_8
Conv1	Kernel size=3×3; Padding=1; Stride=1			
Conv2	Kernel size=3×3; Padding=1; Stride=1			
Conv3	None			
Output size	64×22×22	128×11×11	256×6×6	512×3×3

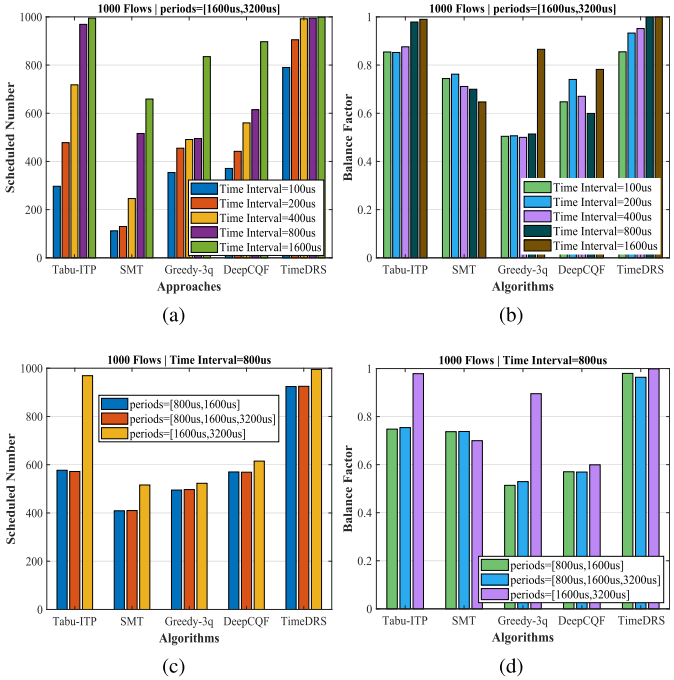
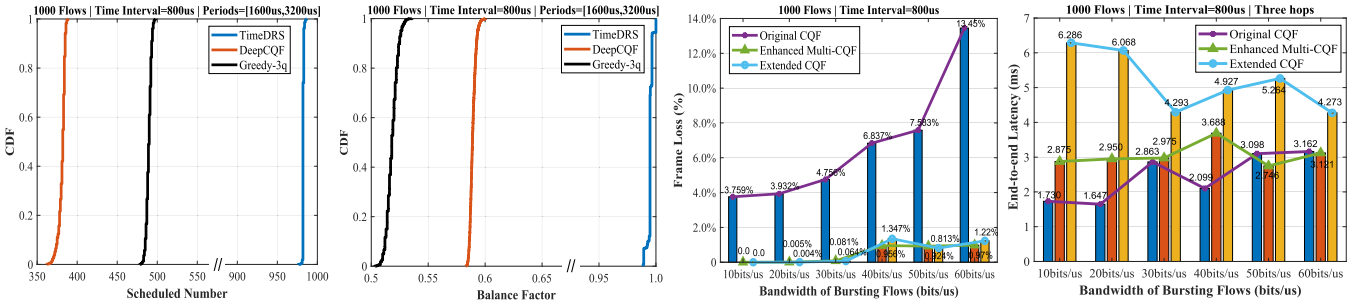


Fig. 9. Evaluation of scheduling performance. (a) Scheduled flow number affected by time intervals; (b) Load balance of various algorithms; (c) Scheduled flow number affected by flow periods; and (d) Load balance under the condition of different flow periods.

Schedulability comparison: Fig. 9(a) shows the TimeDRS can schedule nearly 1,000 flows at most, and increase the number of schedulable TT flows by 2x, 60%, 77.9% and 35% compared with SMT, DeepCQF, Greedy-3q, and Tabu-ITP, respectively, in the condition of $\mathcal{T} = 800us$. Although SMT is very popular in TSN scheduling task, it takes too much time to schedule a large number of flows. While for Greedy-3q, it always selects the current sending time interval with maximal capacity, but it is very hard to predict all the flow aggregation conditions that change in the subsequent forwarding process. Fig. 9(b) depicts the relationship between load balance and scheduling algorithms. The TimeDRS has the best quality in load balance of time interval utilization, as its balance factor deduced by $1-$Eq. (30) can always reach the highest level when setting different time interval sizes. That means TimeDRS can achieve a good trade-off between the schedulable flow number and load balance.

Impact of time intervals: Fig. 9(a) illustrates the relationship between the time interval size and accommodated flow number. As the time interval size gets larger, the number of schedulable TT flows for various algorithms increases mainly for two reasons. One reason is that the number of time intervals within a hyper period decreases according to Eq. (41), which reduces the effort of finding optimal solutions. Another reason is that the capacity for each time interval becomes larger that can accommodate more flows. Fig. 9(b) shows that TimeDRS and Tabu-ITP can better balance the time interval resource utilization, especially when the time interval size is large.

Impact of flow periods: As shown in Fig. 9(c) and Fig. 9(d), we analyze the scheduling performance under the condition of different flow periods, which are set to {800us, 1600us}, {800us, 1600us, 3200us}, and {1600us, 3200us}, respectively. For the flow periods of {800us, 1600us} and {800us, 1600us, 3200us}, the schedulable flow number and load balance are almost the same for different algorithms. This is because the TT flows with shorter periods have the higher generation frequency during a hyper period, and they tend to consume more time interval resources, which significantly affect the performance of scheduling. The flow set with periods {1600us, 3200us} performs the best on schedulable flow number and load balance because the shortest flow period is 1600us, which is larger than the other two flow sets.



(a) (b) (c) (d)

Fig. 10. Experimental results of scheduling online. (a) Cumulative distribution function of scheduled flow number; (b) Cumulative distribution function of load balance; (c) Computational time cost comparison; and (d) Comparison of reliability for bursting flows.

Besides, TimeDRS still has the best scheduling performance on TT flows and load balance compared with SOTA algorithms.

Fig. 10 shows the experimental results of scheduling online. Noting that the computational time cost is required to be in the second level for the incrementally added TT flows, we select TimeDRS, DeepCQF and Greedy-3q as the comparison algorithms to schedule the flows, respectively. The performance analyses are detailed as follows.

On-line schedulability: We totally test 2,000 rounds of scheduling 1,000 TT flows along with bursting flows. For each test round, the frame size of each added flow is randomly assigned from the range of 50B∼1KB [26], and the sequence of added flows is out of order from the flow set. Fig. 10(a) and Fig. 10(b) show the cumulative distribution function (CDF) of the schedulable flow number and balance factor. In particular, TimeDRS can schedule the most number of TT flows with the range of 976∼987, and achieve the best load balance of 0.988∼1.0 compared with Greedy-3q and DeepCQF. Although Greedy-3q can schedule more TT flows than DeepCQF in this case, it does not perform as well on system load balance because it only focuses on allocating the current optimal time intervals for each incoming flow without taking the long-term reward into account.

Loss of bursting flows: In the on-line scheduling scenarios, we need to pay more attention to the bursting flows co-transmitted with TT flows. By controlling the frame generation rate (i.e., bandwidth) of bursting flows, we calculate the frame loss rate within 500 hyper periods (i.e., 2,000 time intervals) by using the original CQF, extended CQF and our enhanced Multi-CQF models, respectively. The extended CQF equipped with two cyclic queues only extends the waiting time of receiving queue to accommodate more flows. Considering the limited resource of each time interval, we gradually increase the bursting flows bandwidth from 10bits/us to 60bits/us, and Fig. 10(d) shows that the frame loss of bursting flows on the original CQF model becomes more severe because of the dead time, setting to 30us on average. In contrast, the frames are barely dropped for the enhanced Multi-CQF and extended CQF models. Specifically, when the bandwidth of bursting flows is up to 60bits/us, the frame loss rate of our enhanced Multi-CQF model can be reduced from 13.45% to 0.97% compared with the original CQF model.

End-to-end latency of bursting flows: As shown in Fig. 10(d), although the reliability of extended CQF performs as good as the enhanced Multi-CQF model, the extended CQF can lead to a very long end-to-end latency for bursting flows. The reason is that the extended CQF model increases the waiting time of cyclic queues to prevent the frames against loss. While for our enhanced Multi-CQF model, it can forward the bursting flows into fast queues to reduce the end-to-end latency. In addition, with the increasing bandwidth of bursting flows, the probability of retransmission in the original CQF model also increases due to frame loss, which enlarges the end-to-end latency and jitter (i.e., latency variance). For simplicity, three hops are simulated, and the enhanced Multi-CQF model reduces the end-to-end latency by almost 2.18x over the extended CQF model.

B. Scheduling on TSN Testbed in Real Environment

Implementations on TSN testbed: In Section VI-A, we evaluated the scheduling performance of TimeDRS based on an assumption that the enhanced Multi-CQF model could guarantee deterministic transmission. In this subsection, we further validate this assumption on a real TSN testbed. The fully centralized architecture is set up on our TSN testbed in Fig. 11(a) as referred to IEEE 802.1Qcc. To support the shaper of our enhanced Multi-CQF model, the NICs on devices are equipped with INTEL I210 that support cables and optical fibers. Detailed by Fig. 11(b), the architecture is composed of two parts, which are control plane for centralized control by the controller, and data plane for transmitting the data flows, respectively. The Centralized User Configuration (CUC) and Centralized Network Configuration (CNC) are the key components in the controller. CUC collects the flow information by interacting with users, and CNC gathers the network status, computes the scheduling, and deploys the configurations. The TimeDRS is implemented in CNC, and we create the database to connect CUC with CNC. To support running TimeDRS algorithm, the high performance hardwares of CPU (AMD R9 5950X), GPU (NVIDIA QUADRO RTX 6000), and 64GB Kingston RAM are utilized in controller on Ubuntu 20.04 with kernel 5.11.0-41-generic. In data plane, we set up the benchmark topology with 10 hosts and 2 TSN switches, each with CPU (INTEL CORE i7-6700K) and 16GB memory, taking the Traffic Control (TC) tool to achieve the

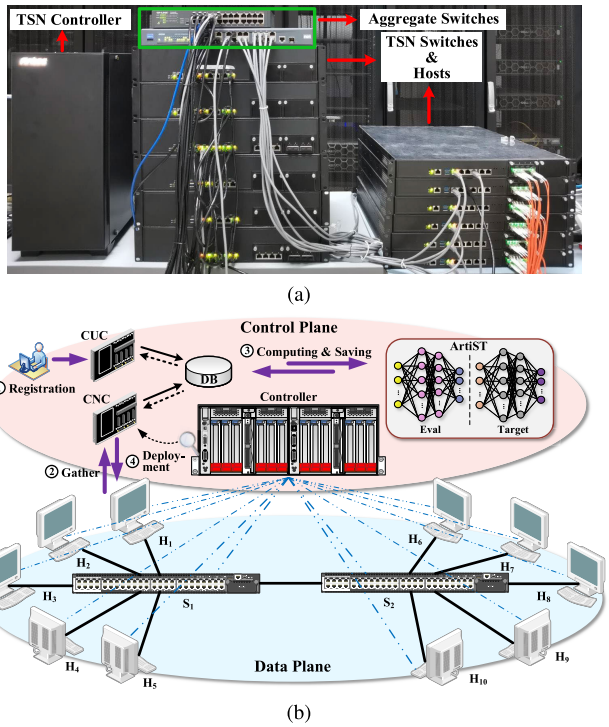


Fig. 11. Architecture of our TSN testbed. (a) Entities of TSN devices; and (b) Fully-centralized architecture of TSN.

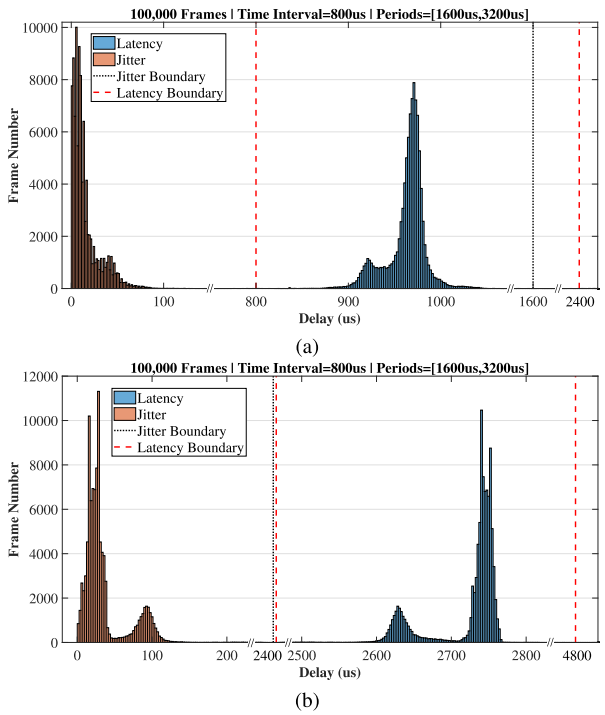


Fig. 12. End-to-end latency and jitter measured on TSN testbed. (a) Deterministic transmission with the original CQF model; and (b) Deterministic transmission with the enhanced Multi-CQF model.

original CQF or Multi-CQF based queuing regularities of TAPRIO. The time synchronization of the global network is achieved by LinuxPTP installed in each device with the same Ubuntu version of controller. The TT flows are generated from SAMPLE-APP-TAPRIO flow generator [27]. The time interval and flow periods are set to 800us and {1600us, 3200us}, respectively. And the other parameters are listed in Table I. The scheduling procedures can be concluded as follows.

Step 1: *Registration.* The TT flows, each with 50B∼1KB frame size are registered to CUC by users, where the source hosts are selected from H_1 to H_5, and destination hosts are chosen from H_6 to H_{10} to ensure they pass through two hops along the path. Then the flow features in Eq. (2) are stored into database of the flow templates.

Step 2: *Gathering.* The network status, including topology and available resources, are detected by CNC in real time through the Link Layer Discovery Protocol (LLDP). Then the status information can be used to update the database.

Step 3: *Computation.* According to the flow features and gathered network resource status in database, TimeDRS computes the allocation policies for mapping each flow into different time intervals.

Step 4: *Deployment.* Based on the computing results extracted from database and parameters we set, CNC deploys the configurations of GCLs to switches, and manages the sending time windows for each TT flow from source hosts.

The end-to-end latency can then be measured by capturing the frames from source and destination hosts. Taking 50 TT flows with 100,000 frames for scheduling as an example, we validate the deterministic flow transmission with the original CQF and our enhanced Multi-CQF models, respectively. Fig. 12(a) shows the scheduling results with the original CQF model. After converging, the average end-to-end latency of CQF model is 963.42us, with the minimum and maximum values of 823.54us and 1595.54us, respectively. Thus, the actual end-to-end latency (823.54us∼1595.54us) of the original CQF model is within the expected theoretical range of 800us∼2400us. The average jitter is 15.17us with a maximum value of 632.12us, which is below the upper limit of 1600us. Moreover, the flow transmission with the enhanced Multi-CQF model is depicted by Fig. 12(b). The end-to-end latency is in the range of 2585.01us∼2882.57us, with the average value of 2711.01us. This is within our expected latency range of 2400us∼4800us, which is computed by Eq. (9). The maximum and average values of jitter are 308.73us and 17.78us, respectively, which satisfies with Eq. (9). As a result, the deterministic performance for transmitting the flows can be ensured, and users can flexibly select the model from the above two CQF models to well balance the frame loss of bursting flows and end-to-end latency of TT flows.

VII. Related Work

With the rising of TSN techniques, considerable works have focused on real-time and high-reliable flow transmission. Steiner et al. [28] concentrated on minimizing the transmission latency of TT flows based on TAS shaper by utilizing the method of SMT. Nevertheless, the time consumed by SMT increases exponentially with the growing number of flows. To reduce the computational time and obtain optimal results, the optimization modulo theories (OMT) for jointly scheduling and routing were further researched by Xu et al. [9]. Yan et al. [11] proposed an injection time planning

(ITP) mechanism based on the Tabu Search algorithm to optimize the temporal scheduling of TT flows with CQF model, and it greatly improved the schedulable flow number and network resource utilization. Taking the jitter-sensitive flows into consideration, Huang et al. [29] presented a time-aware cyclic-queuing (TACQ) mechanism that combines TAS and CQF together to support the co-transmission of TT and isochronous flows. Yuan et al. [30] proposed an adaptive priority adjustment scheduling method with analysis of response time, which improved the successful rate and reduced worst-case end-to-end latency for transmitting TT flows in TSN. As a result, the off-line scheduling tasks in TSN have been well planned.

After the network goes online, dealing with the incrementally added new flows can be a significant challenge. Li et al. [25] presented an enhanced reconfiguration mechanism to avoid the conflicts of various arrival TT flows based on the dependence graph. Quan et al. [31] proposed a dynamic scheduling mechanism based on CQF model with the light-weight framework. To further reduce the transmission complexity of mixed flows, Zhang et al. [32] set up a divisibility theory-based analysis approach that significantly improved both runtime efficiency and network load balance. However, it is difficult for traditional algorithms to fully utilize the available network resources. Hence, the DRL-based resource scheduling algorithms in TSN are further investigated. Prados-Garzon et al. [33] scheduled flows in the asynchronous networks by designing a DRL-based solution LEARNET, which achieved higher gains of revenue in the 5G backhaul environment. Zhou et al. [14] significantly decreased the occurrence of misbehaviors when sending TT flows by utilizing the Deep Deterministic Policy Gradient framework.

Different from the above related works, we focus on scheduling the mixed flows (i.e., TT and bursting flows) in CQF-based TSN. The enhanced Multi-CQF model is proposed to satisfy the QoS demands of bursting flows without affecting TT flow transmission. To fully utilize the large time interval and queuing resources in the queue model, a time-correlated DRL resource scheduling algorithm is further designed that effectively improves scheduling performance of whole system.

VIII. Conclusion

In this paper, we have investigated the mixed-flow (i.e., TT and bursting flows) scheduling problem in CQF-based TSN. We have designed an enhanced Multi-CQF model, which defines the queuing principles and flow receiving strategies to support the deterministic transmission for both bursting and TT flows. To fully utilize the time-related network resources in our queue model, the TimeDRS algorithm has been proposed to jointly schedule the time interval and queuing resources in the unit of flow. Simulation results have demonstrated that our enhanced Multi-CQF model can greatly reduce the frame loss and end-to-end latency of bursting flows. Meanwhile, the TimeDRS can schedule more TT flows and achieve better load balance of the whole system. Different from the original CQF, our enhanced Multi-CQF model is able to guarantee the high-reliable and low-latency transmission of bursting flows while satisfying the deterministic latency, jitter, and high reliability demands of TT flows. The TimeDRS algorithm can also be extended into more generic time-related resource scheduling scenarios. For the future work, we will investigate the reliable transmission for bursting flows in DetNet.

References

[1] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless time sensitive networking impact on an industrial collaborative robotic workcell," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7351–7360, Oct. 2022.

[2] L. Xu et al., "Learning-based scalable scheduling and routing co-design with stream similarity partitioning for time-sensitive networking," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13353–13363, Aug. 2022.

[3] L. Deng, G. Xie, H. Liu, Y. Han, R. Li, and K. Li, "A survey of real-time Ethernet modeling and design methodologies: From AVB to TSN," *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–36, Jan. 2022.

[4] S. Schriegel and J. Jasperneite, "A migration strategy for profinet toward Ethernet TSN-based field-level communication: An approach to accelerate the adoption of converged IT/OT communication," *IEEE Ind. Electron. Mag.*, vol. 15, no. 4, pp. 43–53, Dec. 2021.

[5] M. Kim, D. Hyeon, and J. Paek, "ETAS: Enhanced time-aware shaper for supporting nonisochronous emergency traffic in time-sensitive networks," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10480–10491, Jul. 2022.

[6] *Wireless Systems for Industrial Automation: Process Control and Related Applications*, Standard ISA-100.11a-2011, May 2011.

[7] D. Yang, K. Gong, J. Ren, W. Zhang, W. Wu, and H. Zhang, "TC-Flow: Chain flow scheduling for advanced industrial applications in time-sensitive networks," *IEEE Netw.*, vol. 36, no. 2, pp. 16–24, Mar. 2022.

[8] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 29: Cyclic Queuing and Forwarding*, IEEE Standard 802.1Qch-2017, Jun. 2017.

[9] L. Xu, Q. Xu, Y. Zhang, J. Zhang, and C. Chen, "Co-design approach of scheduling and routing in time sensitive networking," in *Proc. IEEE Conf. Ind. Cyberphys. Syst. (ICPS)*, Jun. 2020, pp. 111–116.

[10] J. Krolikowski et al., "Joint routing and scheduling for large-scale deterministic IP networks," *Comput. Commun.*, vol. 165, pp. 33–42, Jan. 2021.

[11] J. Yan, W. Quan, X. Jiang, and Z. Sun, "Injection time planning: Making CQF practical in time-sensitive networking," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 616–625.

[12] N. Finn. (Sep. 2019). *Multiple Cyclic Queuing and Forwarding*. [Online]. Available: https://www.ieee802.org/1/files/public/docs2019/df-finn-multiple-CQF-0919-v01.pdf

[13] Z. Cheng, D. Yang, W. Zhang, J. Ren, H. Wang, and H. Zhang, "DeepCQF: Making CQF scheduling more intelligent and practicable," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 1–6.

[14] B. Zhou and L. Cheng, "Mitigation of scheduling violations in time-sensitive networking using deep deterministic policy gradient," in *Proc. 4th FlexNets Workshop Flexible Netw. Artif. Intell. Supported Netw. Flexibility Agility*, Aug. 2021, pp. 32–37.

[15] *IEEE Standard for Local and Metropolitan Area Net works—Timing and Synchronization for Time-Sensitive Applications*, IEEE Standard 802.1AS-2020, Jun. 2020.

[16] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L. Wang, "Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, Jun. 2019.

[17] X. Shen et al., "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, 2020.

[18] W. Zhang et al., "Optimizing federated learning in distributed industrial IoT: A multi-agent approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3688–3703, Dec. 2021.

[19] *IEEE Standard for Local and Metropolitan Area Net works—Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, IEEE Standard 802.1Qcc-2018, Oct. 2018.

[20] X. Liao, J. Shi, Z. Li, L. Zhang, and B. Xia, "A model-driven deep reinforcement learning heuristic algorithm for resource allocation in ultra-dense cellular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 983–997, Jan. 2020.

[21] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 1st Quart., 2022.

[22] D. Yang, E. Cui, H. Wang, and H. Zhang, "EH-edge—An energy harvesting-driven edge IoT platform for online failure prediction of rail transit vehicles: A case study of a cloud, edge, and end device collaborative computing paradigm," *IEEE Veh. Technol. Mag.*, vol. 16, no. 2, pp. 95–103, Jun. 2021.

[23] L. Ale, S. A. King, N. Zhang, A. R. Sattar, and J. Skandaraniyam, "D3PG: Dirichlet DDPG for task partitioning and offloading with constrained hybrid action space in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19260–19272, Oct. 2022.

[24] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021.

[25] Z. Li, H. Wan, Z. Pang, Q. Chen, and M. Gu, "An enhanced reconfiguration for deterministic transmission in time-triggered networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1124–1137, Jun. 2019.

[26] Industrial Internet Consortium. (Mar. 2019). *Time Sensitive Networks for Flexible Manufacturing Testbed Characterization and Mapping of Converged Traffic Types*. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_TSN_Testbed_Char_Mapping_of_Converged_Traffic_Types_Whitepaper_20180328.pdf

[27] Intel Corporation. (Apr. 2019). *TSN Reference Software for Linux*. [Online]. Available: https://github.com/intel/iotg_tsn_ref_sw/tree/apollolake-i/

[28] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 42–47, Jun. 2018.

[29] Y. Huang, S. Wang, B. Wu, T. Huang, and Y. Liu, "TACQ: Enabling zero-jitter for cyclic-queuing and forwarding in time-sensitive networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[30] Y. Yuan, X. Cao, Z. Liu, C. Chen, and X. Guan, "Adaptive priority adjustment scheduling approach with response-time analysis in time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8714–8723, Dec. 2022.

[31] W. Quan, J. Yan, X. Jiang, and Z. Sun, "On-line traffic scheduling optimization in IEEE 802.1 Qch based time-sensitive networks," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun., IEEE 18th Int. Conf. Smart City, IEEE 6th Int. Conf. Data Sci. Syst.*, Dec. 2020, pp. 369–376.

[32] Y. Zhang, Q. Xu, L. Xu, C. Chen, and X. Guan, "Efficient flow scheduling for industrial time-sensitive networking: A divisibility theory-based method," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9312–9323, Dec. 2022.

[33] J. Prados-Garzon, T. Taleb, and M. Bagaa, "LEARNET: Reinforcement learning based flow scheduling for asynchronous deterministic networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

Dong Yang (Member, IEEE) received the Ph.D. degree in communication engineering from the School of Electronics and Information Engineering, Beijing Jiaotong University, China, in 2008.

From March 2009 to July 2010, he was a Post-Doctoral Researcher with Jönköping University and ABB Corporate Research, Sweden, where he participated in the standardization and innovation of industrial wireless sensor network. He is currently a Full Professor in communication engineering and the Director of the Intelligent Industry Network Research Institute, Beijing Jiaotong University. His current research interests include network architecture, the Internet of Things, wireless sensor networks, and data driven network resource management. He has authored or coauthored over 70 scientific publications in the field of network and communication technologies. More than 20 of his invention patents were licensed to ZTE. He has served as the Co-Chair for IEEE INFOCOM 2023 Workshop PerAI-6G, the Session Chair for IEEE ICCT 2022, and a TPC Member for IEEE MetaCom 2023. He has been serving as an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING and an Executive Editor for *Transactions on Emerging Telecommunications Technologies* (Wiley). He has served as the Guest Editor for IEEE TRANSACTION ON INDUSTRIAL INFORMATICS in 2021.

Zongrong Cheng (Student Member, IEEE) received the B.E. degree from the School of Automation Engineering, Northeast Electric Power University, Jilin, China, in 2019. He is currently pursuing the Ph.D. degree in communication and information systems with Beijing Jiaotong University, Beijing, China. His current research interests include the Industrial Internet of Things, deterministic networking, resource management, data analytics, and deep reinforcement learning.

Weiting Zhang (Member, IEEE) received the Ph.D. degree in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2021. From November 2019 to November 2020, he was a Visiting Ph.D. Student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. Since December 2021, he has been an Associate Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include the Industrial Internet of Things, deterministic networking, edge intelligence, and machine learning for wireless networks.

Hongke Zhang (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively. From 1992 to 1994, he was a Post-Doctoral Researcher with Beijing Jiaotong University, Beijing, China, where he is currently a Professor with the School of Electronic and Information Engineering and the Director of the National Engineering Laboratory on Next Generation Internet Technologies. He has authored more than ten books and the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He is the Chief Scientist of the National Basic Research Program of China (973 Program) and has also served on the editorial boards for several international journals.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include network resource management, wireless network security, the Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award from IEEE, Canada, in 2019, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award from the IEEE Vehicular Technology Society in 2018, the Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee in 2019 and AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award from the University of Waterloo in 2006 and the Premier's Research Excellence Award (PREA) from the Province of Ontario, Canada, in 2003. He served as the Technical Program Committee Chair/the Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President Elect of the IEEE Communications Society. He was the Vice President for Technical & Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, *IEEE Network*, and *IET Communications*.