# Intelligent End-to-End Deterministic Scheduling Across Converged Networks

Zongrong Cheng, *Member, IEEE*, Weiting Zhang, *Member, IEEE*, Dong Yang, *Member, IEEE*, Chuan Huang, *Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Sherman Shen, *Fellow, IEEE*

***Abstract*—Deterministic network services play a vital role for supporting emerging real-time applications with bounded low latency, jitter, and high reliability. The deterministic guarantee is penetrated into various types of networks, such as 5G, WiFi, satellite, and edge computing networks. From the user's perspective, the real-time applications require end-to-end deterministic guarantee across the converged network. In this paper, we investigate the end-to-end deterministic guarantee problem across the whole converged network, aiming to provide a scalable method for different kinds of converged networks to meet the bounded end-to-end latency, jitter, and high reliability demands of each flow, while improving the network scheduling QoS. Particularly, we set up the global end-to-end control plane to abstract the deterministic-related resources from converged network, and model the deterministic flow transmission by using the abstracted resources. With the resource abstraction, our model can work well for different underlying technologies. Given large amounts of abstracted resources in our model, it is difficult for traditional algorithms to fully utilize the resources. Thus, we propose a deep reinforcement learning based end-to-end deterministic-related resource scheduling (E2eDRS) algorithm to schedule the network resources from end to end. By setting the action groups, the E2eDRS can support varying network dimensions both in horizontal and vertical end-to-end deterministic-related network architectures. Experimental results show that E2eDRS can averagely increase 1.33x and 6.01x schedulable flow number for horizontal scheduling compared with MultiDRS and Multi-Naive algorithms, respectively. The E2eDRS can also optimize 2.65x and 3.87x server load balance than MultiDRS and MultiNaive algorithms, respectively. For vertical scheduling, the E2eDRS can still perform better on schedulable flow number and server load balance.**

***Index Terms*—Deterministic guarantee, end-to-end deterministic-related network architecture, deep reinforcement learning.**

## I. Introduction

In recent few years, deterministic network technologies have aroused great attention [1], [2], [3]. To promote the development of deterministic network technologies, time-sensitive networking (TSN) is proposed by IEEE 802.1 TSN task group that aims to guarantee the bounded end-to-end latency, jitter, and high reliability at the layer 2 of network [4], [5], [6]. To further enlarge the scale of deterministic network services, the Internet Engineering Task Force (IETF) working group proposes the deterministic networking (DetNet), which operates over layer 2 bridged and layer 3 routed segments [7], [8].

Unfortunately, the existing TSN and DetNet technologies are too monotonous to satisfy with the increasing demands of diversified real-time applications, such as autonomous driving, remote control, among others [9], [10]. To meet the differentiated quality of service (QoS) demands in various applications, the deterministic guarantee is penetrated into more types of networks. To enhance the determinacy of wireless networks, Li et al. [11] proposed a decentralized flow scheduling method to intelligently make packet transmission decisions for collision-free 5G systems. Jiang et al. [12] developed a deterministic satellite network transmission approach to predict all upcoming communication opportunities and construct deterministic routing paths. Sudhakaran et al. [13] presented a method to configure the WiFi-enabled TSN and meet the application time budget. In multi-access edge computing networks, Peng et al. [14] proposed a task deterministic network architecture to ensure the bounded flow transmission latency and zero jitter for critical tasks. As such, each type of network has its own architecture and underlying technologies to enable the deterministic services.

However, from the user's point of view, the real-time applications need end-to-end deterministic guarantee across the converged network, which is composed by multiple heterogeneous networks [2], [15], [16]. Unluckily, the convergence of different deterministic networks is still at the early stage, existing approaches for deterministic guarantee are restricted to their focused type of network and study each network in isolation [17], [18], [19]. None of the related works consider a scalable method to ensure the end-to-end determinacy over the whole converged network. Inspired by increasing the practicability, we investigate the end-to-end deterministic guarantee problem across the whole

Received 7 November 2023; revised 15 May 2024; accepted 3 September 2024. Date of publication 16 January 2025; date of current version 6 March 2025. 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 62425104, in part by the Shanghai Sailing Program under Grant 24YF2717400. Recommended for acceptance by L. Bai. *(Corresponding author: Weiting Zhang.)*

Zongrong Cheng is with Shanghai Aerospace Electronic Technology Institute, Shanghai 201109, China (e-mail: zrcheng@bjtu.edu.cn).

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

Chuan Huang is with the Future Network of Intelligence Institute (FNii), School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: huangchuan@cuhk.edu.cn).

Xuemin Sherman 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/TMC.2025.3530486

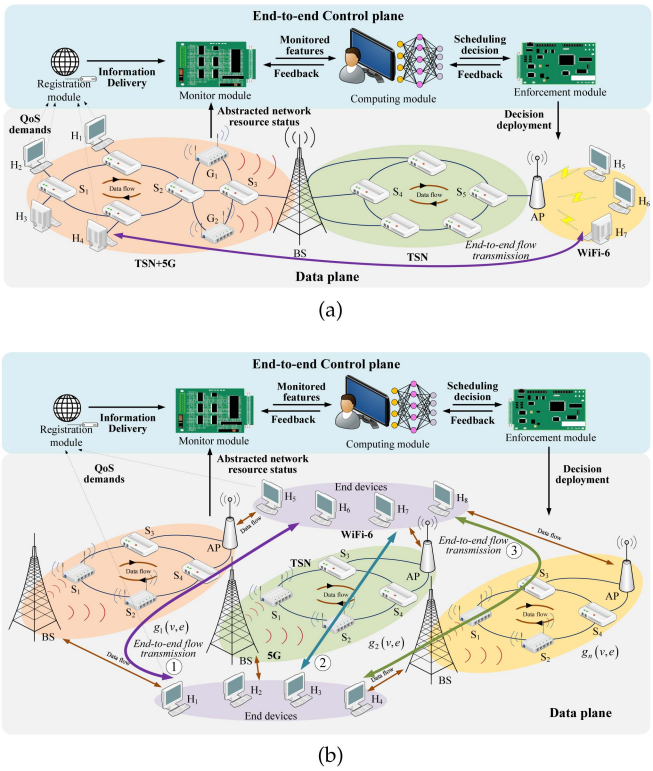


Fig. 1. Horizontal and vertical end-to-end deterministic-related network architectures. (a) horizontal network architecture; and (b) vertical network architecture.

converged network, aiming to provide a scalable method for different kinds of converged networks to satisfy with the bounded end-to-end latency, jitter, and high reliability requirements of each flow, while improving the network scheduling QoS.

In this paper, we introduce two types of network architectures for network converging, namely horizontal and vertical end-to-end deterministic-related network architectures. Fig. 1(a) shows the horizontal end-to-end deterministic-related network architecture [20], which is composed by heterogeneous deterministic networks inter-connected with each other, such as 5G+TSN+WiFi-6. The vertical end-to-end deterministic-related architecture [21] shown in Fig. 1(b) is formed by multiple paralleled horizontal networks. In this architecture, the network with higher communication quality can be selected to transmit the flows so that the end-to-end latency, throughput, etc. can be optimized. Moreover, the multiple paralleled networks can serve as redundancies to enhance the flow transmission reliability.

Nevertheless, it is of great challenge for off-the-shelf isolated approaches to support the end-to-end converged network determinacy both in horizontal and vertical end-to-end deterministic-related network architectures. First, because the underlying technologies of heterogeneous deterministic networks are very different [15], [22], the isolated approaches for each type of network cannot be scaled into the whole converged network. Therefore, the network performance and scalability are degraded. In addition, large number of isolated approaches in control plane need to repeatedly negotiate with each other [23], [24]. This negotiation process can significantly increase the computational time and energy cost. Hence, there is a great need for converged network to scale and unify the approaches. Moreover, existing approaches only focus on time slot and frequency resources to support determinacy. It cannot meet the resource scheduling requirement of converged network with more deterministic-related resources, such as computing, energy, and memory resources.

Based on the commonality of network properties, we overcome the challenges by managing the abstracted resources from converged network. In specific, we set up a global end-to-end control plane to manage the flow transmission in different network architectures from end to end. To guarantee the determinacy, the end-to-end deterministic-related resource abstraction model is established by leveraging the abstracted deterministic-related resources from monitor module in control plane. The network resource abstraction can effectively eliminate the gap caused by different underlying network technologies. As such, the end-to-end deterministic-related resource abstraction model is not restricted to any fixed networks. Extended from the existing deterministic technologies that focus on time slot and frequency resources, we also jointly consider more resources in the model, such as routing, power, computing, memory, and energy resources.

To optimize the network QoS, the abstracted resources in our model need to be scheduled. The resource scheduling in single deterministic network has been proved as an NP-hard [3], [17]. Thus, the multiple network scheduling in our architecture could be more complex. Traditional scheduling algorithms, such as dynamic programming and heuristics, are too computational time cost and lack of flexibility [25]. To improve the resource scheduling efficacy, we propose a deep reinforcement learning (DRL) based end-to-end deterministic-related resource scheduling (E2eDRS) algorithm. Different from the existing DRL-based scheduling algorithms that only schedule the available resources in their isolated and fixed networks [23], [26], the E2eDRS embeds with an end-to-end resource consumption matrix to focus on end-to-end deterministic-related resource scheduling over converged network flexibly composed by heterogeneous networks. This matrix can also assist E2eDRS extending into more generic time-related network resource scheduling scenarios, such as joint time-frequency domain scheduling, or other types of time division multiple access (TDMA) based resource scheduling in converged network. In addition, by setting up the action groups, the E2eDRS can support varying network dimensions both in horizontal and vertical end-to-end deterministic-related network architectures.

The scheduling performance is evaluated in horizontal and vertical end-to-end deterministic-related network architectures with the benchmark 5G+TSN converged network topologies. For horizontal scheduling, our E2eDRS algorithm can averagely increase 1.33x and 6.01x schedulable flow number compared with MultiDRS and MultiNaive algorithms, respectively. The E2eDRS can also optimize 2.65x and 3.87x server load balance than MultiDRS and MultiNaive, respectively. For vertical scheduling, the E2eDRS still performs better on schedulable flow number and server load balance compared with MultiDRS and MultiNaive algorithms. Moreover, E2eDRS makes a good trade-off between computational time cost and scheduling

ability. These experimental results indicate that the global end-to-end deterministic scheduling performs better than the independent optimization of each network. The contributions of this paper are as follows:

- *Horizontal and vertical end-to-end deterministic-related network architectures:* We present horizontal and vertical end-to-end deterministic-related network architectures for network converging. The horizontal and vertical architectures can abstract the deterministic-related resources from horizontal and vertical scheduling scenarios, respectively.
- *End-to-end deterministic-related resource abstraction model:* We establish the end-to-end deterministic-related resource abstraction model to guarantee the end-to-end determinacy in horizontal and vertical end-to-end deterministic-related network architectures. With the resource abstraction, the model works well for heterogeneous deterministic networks. The routing, time slot, spectrum, power, computing, and memory resources are jointly considered by this model.
- *DRL-based end-to-end deterministic-related resource scheduling algorithm:* We propose the DRL-based end-to-end deterministic-related resource scheduling algorithm, namely E2eDRS, to effectively schedule a large number of abstracted deterministic-related resources in our model. Embedded with an end-to-end resource consumption matrix, the E2eDRS can focus on end-to-end scheduling across the converged network. E2eDRS can also be used in generic time-related resource scheduling scenarios, such as TDMA-based resource scheduling. The action groups in E2eDRS support varying network dimensions in different architectures.

The remainder of this paper is organized as follows. Section II presents horizontal and vertical end-to-end deterministic-related network architectures for deterministic network converging. To guarantee the network determinacy, the end-to-end deterministic-related resource abstraction model is also established. Section III formulates the multi-objective optimization problem for scheduling deterministic-related resources in our model, and transforms the problem into a Markov Decision Process (MDP). Section IV proposes the E2eDRS algorithm to schedule the deterministic-related resources from end to end. Finally, Section VI concludes this paper.

## II. The End-to-End Deterministic-Related Resource Abstraction Model

This section proposes the end-to-end deterministic-related resource abstraction model that supports both horizontal and vertical end-to-end deterministic-related network architectures. In this model, different network resources, including routing, time slot, spectrum, power, computing, and memory resources are abstracted from the heterogeneous networks to jointly guarantee the deterministic flow transmission from end to end. With the abstracted resources and flow features of the model, the deterministic guarantee is not restricted to any specific networks, such as TSN, DetNet or 5G+TSN. Any inter-connected networks that provide deterministic transmission related resources can be jointly planned. The model focuses on end-to-end network performance when the flow goes through different networks.

### A. Horizontal and Vertical End-to-End Deterministic-Related Network Architectures

In real industrial scenario, different networks are converged in horizontal and vertical forms impacting scheduling performance. Hence, the end-to-end deterministic-related resource abstraction model needs to support both horizontal and vertical end-to-end deterministic-related network architectures. The two architectures are introduced as follows.

Fig. 1(a) shows the horizontal end-to-end deterministic-related network architecture. In data plane, different networks, such as TSN, 5G, and WiFi-6, connect with each other, and they converge into one horizontal network denoted as $net_1$. The data flow can be transmitted across $net_1$ based on the deployed deterministic scheduling policy. The control plane globally schedules the deterministic-related resources for horizontal network from end to end. In specific, the registration module collects the flow information and QoS demands generated by users, and delivers these information to the monitor module. The monitor module also detects the network resource status and system feedbacks through signaling. The detected information by monitor module is then encapsulated into the monitored features and passed to the computing module. To globally schedule the flows from end to end, the computing module observes the monitored features and jointly infers the deterministic-related resource scheduling policies based on optimization goals. Then, the computed policies are deployed into the network by enforcement module. Network protocols, such as NETCONF, can be used to configure the network devices, and the configuration results are fed back to further improve the training performance.

Moreover, multiple paralleled horizontal networks can compose a vertical network. The vertical end-to-end deterministic-related network architecture is depicted by Fig. 1(b). In data plane, the paralleled horizontal networks are denoted as $net_1$, $net_2, \ldots, net_n$, respectively. When a new flow arrives, a better network $net_\epsilon$ can be selected for flow transmission by the controller. By this way, the scheduling QoS can be further improved. Let the undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be the physical network topology. $\mathcal{V}$ and $\mathcal{E}$ denote a set of nodes and links, respectively. $\mathcal{G}(\mathcal{V}, \mathcal{E})$ includes all nodes and links in the network, thus

$$\mathcal{G}(\mathcal{V}, \mathcal{E}) = \{v_i, [v_i, v_k] | \forall v_i \in \mathcal{V}, \forall [v_i, v_k] \in \mathcal{E}, v_i \neq v_k\}. \quad (1)$$

$\mathcal{G}(\mathcal{V}, \mathcal{E})$ consists of multiple paralleled horizontal networks with topology $g_\epsilon(v, e)$. As a special case, when the network architecture is horizontal, then $g_1(v, e) \cong \mathcal{G}(\mathcal{V}, \mathcal{E})$. Each flow $f_j \in \mathcal{F}$ going through the network is featured by flow ID, source address, destination address, packet size, packet number, latency and jitter requirements, etc. These flow features can be described by

$$f_j = \left\{id_j, src_j, dst_j, size_j, num_j, \mathcal{D}_j^{req}, \mathcal{J}_j^{req}\right\}, \quad (2)$$

where $j = 1, 2, \ldots, |\mathcal{F}|$, and $|\mathcal{F}|$ is the total number of flows. As a big shining point, the end-to-end deterministic scheduling mechanism in control plane can also be scaled into our vertical end-to-end deterministic-related network architecture. Hence, we focus on the design of deterministic-related resource scheduling mechanism for both horizontal and vertical networks in control plane.

### *B. End-to-End Deterministic-Related Resource Abstraction Model*

To ensure the deterministic flow transmission across the horizontal and vertical end-to-end deterministic-related network architectures, the end-to-end deterministic-related resource abstraction model is set up by using the monitored features. End-to-end bounded flow transmission latency, jitter, and high reliability are the core indicators of deterministic flow transmission. The end-to-end latency refers to the summation of flow transmission delay on each node and link along its path. Let $\mathcal{D}_j^{(v_i)}$ and $\mathcal{D}_j^{(\ell)}$ denote the delay on each node $v_i$ and link $\ell$, respectively. The $\ell$ is the link between adjacent nodes $[v_i, v_{i+1}]$. Hence, the end-to-end latency $\mathcal{D}_j$ can be described by

$$\mathcal{D}_j = \sum_{i=1}^{\mathcal{N}} \mathcal{D}_j^{(v_i)} + \sum_{\ell=1}^{\mathcal{L}} \mathcal{D}_j^{(\ell)}, \tag{3}$$

where $\mathcal{N}$ and $\mathcal{L}$ are the number of nodes and links that the flow goes through. The jitter is end-to-end latency variance of each flow. Let $\mathcal{D}_j^{(k)}$ be the end-to-end latency of flow $f_j$ on packet $k$. Hence, the flow jitter $\mathcal{J}_j$ is the difference between maximal and minimal packet latency, described by

$$\mathcal{J}_j = \left| \max\left(\mathcal{D}_j^{(k)}\right) - \min\left(\mathcal{D}_j^{(k)}\right) \right|. \tag{4}$$

The high reliability requires zero packet loss of each flow. Thus, the rate of packet loss should be measured. Let function $\phi(\cdot)$ indicate whether the flow can be successfully scheduled. $\phi(\cdot) = 1$ indicates the flow is successfully scheduled, otherwise, $\phi(\cdot) = 0$. Then, the packet loss rate is described by

$$\mathcal{R}_j = \frac{\left| num_j - \sum_{k=1}^{num_j} \phi\left(f_j^{(k)}\right) \right|}{num_j} \times 100\%, \tag{5}$$

where $f_j^{(k)}$ is the $k$-th frame of flow $f_j$.

In real systems, end-to-end deterministic flow transmission can be ensured by using different underlying technologies across heterogeneous networks, such as queuing gates, 5G TSN-translators, etc. However, the differentiated underlying technologies involving physical, MAC, network, and higher layers bring the difficulty for network inter-connection and end-to-end modeling. Fortunately, these technologies commonly aim to schedule the deterministic-related resources to ensure the determinacy. To overcome the challenge and focus on end-to-end modeling, we abstract deterministic-related resources from $\mathcal{G}(\mathcal{V}, \mathcal{E})$ supported by the underlying technologies in different layers. As such, the end-to-end deterministic flow transmission is guaranteed by allocating the abstracted resources in essence. Extended from existing traffic shaping technologies that mainly consider time slot and frequency resources, our abstracted resources not only involve routing, time slot, frequency, and power for flow transmission, but also embrace the computing, energy, and memory resources at each node for deterministic processing.

Routing resource is the flow path allocated to $f_j$. Along the flow path, the requirements of bounded transmission latency, jitter, and high reliability need to be satisfied. To avoid increasing the latency and jitter, the path with no-loop is required. The $\mathcal{D}_j^{(\ell)}$ in (3) maps the allocated time slot resource on link $\ell$. To guarantee the high reliable transmission during the transmission time $\mathcal{D}_j^{(\ell)}$, the model requires that the flow $f_j$ should be accommodated within $\mathcal{D}_j^{(\ell)}$. And the occupied time slot size of $\mathcal{D}_j^{(\ell)}$ cannot exceed the upper limit $\mathcal{C}_\ell$. Hence,

$$size_j \le \mathcal{B}_j^{(\ell)} \times \mathcal{D}_j^{(\ell)} \le \mathcal{C}_\ell, \tag{6}$$

where $\mathcal{B}_j^{(\ell)}$ is the link bandwidth that flow $f_j$ occupied. $\mathcal{C}_\ell$ refers to the maximal link capacity in time slot $\mathcal{D}_j^{(\ell)}$. Considering wireless networks, both the time slot and frequency bandwidth resources can affect the deterministic transmission. Let $\gamma$ be the signal to interference noise ratio. To accommodate flow $f_j$ during the transmission time $\mathcal{D}_j^{(\ell)}$, the model requires

$$size_j \le \mathcal{B}_j^{(\ell)} \times \log_2(1+\gamma) \times \mathcal{D}_j^{(\ell)} \le \mathcal{C}_\ell, \tag{7}$$

where $\mathcal{B}_j^{(\ell)}$ allocated to flow $f_j$ is the channel bandwidth. The $\gamma$ reflects the quality of received signal. Specifically, $\gamma$ is determined by power resource for flow transmission. Let $p_\ell$ be the transmission power on link $\ell$. To maintain a high communication quality, it requires that the value of $\gamma$ is no less than the threshold $\gamma_0$. Thus,

$$\gamma = \frac{p_\ell \times g_{\ell\ell}}{1 + \sum_{\ell'} p_{\ell'} \times g_{\ell'\ell}} \ge \gamma_0, \quad \ell' \ne \ell, \tag{8}$$

where $g_{\ell\ell}$ is the channel gain of link $\ell$, and $g_{\ell'\ell}$ is the gain of interference from link $\ell'$ to $\ell$.

Except for link transmission delay, the processing delay $\mathcal{D}_j^{(v_i)}$ on each node also impacts the end-to-end deterministic flow transmission. For the computation-intensive tasks carried by each flow, the $\mathcal{D}_j^{(v_i)}$ maps the computational time cost on the node. Let $\mu_j \in [0, 1]$ indicate the ratio of CPU frequency allocated to $f_j$. CPU frequency $\mathcal{Q}_{v_i}$ means the amount of data that node $v_i$ can be processed per unit time. The $\vartheta$ denotes the number of CPU cycles used for dealing with one bit data. To complete the computation task within $\mathcal{D}_j^{(v_i)}$ and guarantee the high reliability on each node, the model requires that the processed data during $\mathcal{D}_j^{(v_i)}$ should be no smaller than task size $size_j^{(v_i)}$ offloaded on node $v_i$. Hence, we have

$$\frac{\mu_j \times \mathcal{Q}_{v_i} \times \mathcal{D}_j^{(v_i)}}{\vartheta} \ge size_j^{(v_i)}. \tag{9}$$

The computation task consumes the energy $E_{v_i}$ on each node. To maintain system operation, the consumed energy should be constrained within a certain range. Therefore,

$$E_{v_i} = \frac{\kappa}{2} \vartheta \times size_j^{(v_i)} \times (\eta \times \mu_j)^2 \le E_{max}^j, \tag{10}$$

where $\kappa$ and $\eta$ are efficiency coefficients. The energy threshold for dealing with $f_j$ is denoted as $E^j_{max}$. The memory resource $\mathcal{M}_{v_i}$ on node $v_i$ stores the flow. To avoid the packet loss on each node, the memory allocated to $f_j$ should be no smaller than the required size $size_j^{(v_i)}$. Thus,

$$\mathcal{M}_{v_i} \times \chi_j \geq size_j^{(v_i)}, \tag{11}$$

where $\chi_j \in [0, 1]$ is the ratio of memory resource allocated to $f_j$. Other resource constraints can also be flexibly added for end-to-end deterministic transmission guarantee.

## III. Problem Formulation and Transformation

This section formulates the multi-objective optimization problem based on the end-to-end deterministic-related resource abstraction model. The complexity of solving the problem is analyzed, and then the problem is transformed into a Markov Decision Process (MDP).

### A. Multi-Objective Optimization Problem Formulation

In the end-to-end deterministic-related resource abstraction model, large number of abstracted deterministic-related resources need to be scheduled to further optimize the end-to-end network QoS. Thus, we have the multi-objective optimization problem $\mathcal{P}_0$, aiming to improve the QoS, such as schedulable flow number, load balance, etc. The optimization goal can be generally described by function $\mathscr{F}(\cdot)$. The decision variables in the function are the allocated deterministic-related resources for each flow $f_j$. The end-to-end deterministic-related resource abstraction model can provide different deterministic-related resources abstracted from the converged network described by $\mathscr{A}_i(g_\epsilon(v, e))$, including routing, time slot, frequency, power, computing, memory, and energy resources, among others. The deterministic flow transmission latency, jitter, and high reliability demands should be also satisfied in $\mathcal{P}_0$. Therefore, the optimization problem $\mathcal{P}_0$ can be formulated by

$$\mathcal{P}_0 : Max(Min) \quad \mathscr{F}\{f_j.\mathscr{A}_i(g_\epsilon(v, e))\} \tag{12}$$

$$s.t. \quad \mathcal{D}_j \leq \mathcal{D}_j^{req}, \tag{12a}$$

$$\mathcal{J}_j \leq \mathcal{J}_j^{req}, \tag{12b}$$

$$\mathcal{R}_j \approx 0, \tag{12c}$$

$$j = 1, 2, \ldots, |\mathcal{F}|, \tag{12d}$$

$$(6), (7), (8), (9), (10), \text{and } (11). \tag{12e}$$

The (12a) requires that real end-to-end transmission latency $\mathcal{D}_j$ does not exceed the flow latency demand $\mathcal{D}_j^{req}$. To satisfy with the jitter requirement, the (12b) shows that the latency variance of flow cannot exceed the threshold $\mathcal{J}_j^{req}$. The (12c) requires near zero packet loss to guarantee the high reliable flow transmission. Constrained by (12d), each flow $f_j$ should belong to the flow set $\mathcal{F}$. The constraints (6), (7), (8), (9), (10), and (11) guarantee the proper resource scheduling for the converged network.

To solve the problem $\mathcal{P}_0$, the effective deterministic-related resource scheduling algorithm needs to be further designed according to the problem complexity analysis.

### B. Problem Analysis

To design an effective algorithm to solve the problem, the complexity of this problem needs to be analyzed.

*Theorem 1:* The multi-objective optimization problem $\mathcal{P}_0$ described by (12) is an NP-hard.

*Proof:* We use the reduction method to prove Theorem 1. The core idea of reduction method is to find a special case of the optimization problem that can be mapped into an existing well-known NP-hard problem. Specifically, we focus on the time slot resource allocation, and set the fixed parameters to other allocated resources. When the time slot number is one, whose length is $D_j^\ell$, the problem is transformed to decide which flow should be injected into this time slot, and the corresponding revenue can be obtained. Let $x_j$ indicate whether flow $f_j$ injects into the time slot, and $r_j$ is the revenue of such an operation. Thus, the optimization goal of (12) is to maximize the total revenue, where

$$Max \quad \sum_{j=1}^{|\mathcal{F}|} r_j \times x_j. \tag{13}$$

Deduced by the time slot resource constraint of (6), the occupied capacity of all the flows during $D_j^\ell$ cannot exceed the maximal link capacity. Hence, it is subject to

$$\begin{cases} \mathcal{D}_j^{(\ell)} \sum_{j=1}^{|\mathcal{F}|} \left( \mathcal{B}_j^{(\ell)} \times x_j \right) \leq \mathcal{C}_\ell, \\ x_j \in \{0, 1\}. \end{cases} \tag{14}$$

The above model is the same with 0–1 knapsack problem, which has been proved as NP-hard. Hence, we prove that the 0–1 knapsack problem can be reduced to an instance of ours. This completes the proof. □

*Theorem 2:* The multi-objective optimization problem $\mathcal{P}_0$ described by (12) is non-convex.

*Proof:* First, the routing resource constraint of the end-to-end deterministic-related resource abstraction model is the integer constraint because the decision variable of flow path is an integer. The variables of other allocated resources in (6)-(11) are non-negative real numbers. Second, the non-linear constraint, such as (10), exists in the model. Therefore, the end-to-end deterministic-related resource abstraction model describes a mixed-integer non-linear programming problem (MINLP), which has been proved as non-convex [27]. This completes the proof. □

Considering the complexity of this NP-hard and non-convex problem, the traditional algorithms, such as dynamic programming and heuristics, spend too much time to compute the resource allocation results, and they are difficult to adapt to dynamic flow arrival. To effectively learn the relationship between each time slot and deterministic-related resources, we design a DRL-based resource scheduling algorithm. With our designed shining points, the multi-objective optimization problem $\mathcal{P}_0$ can be well-solved.

### C. Problem Transformation

In DRL, the agent learns to perform tasks by interacting with environment. At each time step, the agent observes state of environment and makes decisions based on the policy provided by neural network. The state is then transited and reward is fed back in terms of Markov Decision Process (MDP) [28], [29], [30]. The objective of learning is to optimize the expected cumulative reward by updating the policy. As the basis of DRL algorithm, we transform the multi-objective optimization problem $\mathcal{P}_0$ into an MDP. The key point of MDP is to build the observation space, action space, and reward [4], [31]. In state transition process, each incrementally added flow utilizes enough deterministic-related resources to ensure the end-to-end deterministic transmission latency, jitter, and high reliability. Therefore, the observation space $\mathcal{O}$ should be composed by the deterministic-related resource utilization $\mathcal{O}_{net}$ and incrementally added flow features at each time step, which can be described by

$$\mathcal{O} = \mathcal{O}_{net} \cup \mathcal{O}_{flow}. \tag{15}$$

According to deterministic-related resource constraints, including link capacity, CPU frequency, energy, and memory resource constraints in problem $\mathcal{P}_0$, the key elements of $\mathcal{O}_{net}$ should be composed by the utilization of link capacity $\mathscr{H}_{\mathcal{C}_\ell}$, CPU frequency $\mathscr{H}_{\mathcal{Q}_{v_i}}$, computation energy $\mathscr{H}_{E_{v_i}}$, and memory space $\mathscr{H}_{\mathcal{M}_{v_i}}$. The state is transited with the changing of resource utilization at each time step [32]. Thus, we have

$$\mathcal{O}_{net} = \left\{\mathscr{H}_{\mathcal{C}_\ell}, \mathscr{H}_{\mathcal{Q}_{v_i}}, \mathscr{H}_{E_{v_i}}, \mathscr{H}_{\mathcal{M}_{v_i}}\right\}, \tag{16}$$

and each element in $\mathcal{O}_{net}$ is the resource utilization in the range of 0∼1. The $\mathcal{O}_{flow}$ comprises each flow features described by (2), which is

$$\mathcal{O}_{flow} = \left\{id_j, src_j, dst_j, size_j, num_j, \mathcal{D}_j^{req}, \mathcal{J}_j^{req}\right\}. \tag{17}$$

The deterministic flow transmission relies on the deterministic-related resource scheduling. Therefore, the action space $\mathcal{A}$ consists of the deterministic-related resource scheduling decisions for each flow $f_j$, which is described by the decision variables $f_j.\mathscr{A}_i(g_\epsilon(v,e))$ in the optimization function. Specifically, the abstracted resources of $f_j.\mathscr{A}_i(g_\epsilon(v,e))$ include the flow path $pth_j$, time slot $\mathcal{D}_j$, frequency bandwidth $\mathcal{B}_j$, transmission power $p_j$, frequency ratio $\mu_j$, and memory ratio $\chi_j$. Thus,

$$\mathcal{A} = \left\{pth_j, \mathcal{D}_j, \mathcal{B}_j, p_j, \mu_j, \chi_j\right\}. \tag{18}$$

The design of reward function relies on the optimization goal described by (12). The more the goal is optimized, the better the reward can be obtained. To improve the efficiency of convergence, the reward shaping method is utilized to build the reward function. The reward shaping measures the gap between current and target scheduling performance in two consecutive steps. Thus, the reward function is described by

$$\zeta_{t+1} = |\mathscr{F}_{t+1} - \Xi| - \beta|\mathscr{F}_t - \Xi|, \tag{19}$$

where the coefficient $\beta \in [0,1]$. The $\mathscr{F}_t$ is the real performance calculated by (12) at step $t$, and $\Xi$ is the ideal status. Various goals can be mapped into (19). For example, to maximize the schedulable number of flows, $\mathscr{F}_t$ should be the successfully scheduled flow number at step $t$. Thus, we have

$$\mathscr{F}_t = \sum_{j=1}^{|\mathcal{F}|} \phi(f_j), \tag{20}$$

and $\Xi$ is the total flow number $|\mathcal{F}|$. For the task flows with high computing power demand, the optimization goal tends to improve the load balance of offloaded task sizes in the distributed servers. We use the standard deviation to reflect the load balance. Thus, the load balance at step $t$ is

$$\mathscr{F}_t = \sqrt{\frac{1}{\mathcal{N}_s}\sum_{i=1}^{\mathcal{N}_s}\left(\mathcal{Q}_{v_i}\sum_{j=1}^{|\mathcal{F}|}\mu_j - \frac{1}{\mathcal{N}_s}\sum_{k=1}^{\mathcal{N}_s}\sum_{j=1}^{|\mathcal{F}|}\mu_j \times \mathcal{Q}_{v_k}\right)^2}, \tag{21}$$

where $\mathcal{N}_s$ is the number of computing servers, and $\Xi \cong 0$ is the ideal status of load balance. When we have $\xi$ optimization goals, the total reward is $\zeta_{t+1} = \sum_{n=1}^{\xi} \zeta_{t+1}^{(n)}$, where $\zeta_{t+1}^{(n)}$ is the shaping reward for each goal.

## IV. DRL-Based End-to-End Deterministic-Related Resource Scheduling Algorithm

This section introduces our shining points and working mechanism of the E2eDRS algorithm implemented in the computing module. In both horizontal and vertical end-to-end deterministic-related network architectures, E2eDRS can intelligently schedule the deterministic-related resources from end to end. As such, the end-to-end deterministic flow transmission can be guaranteed, and the optimization problem can be well-solved.

### A. Shining Points

For simplicity, the one-dimensional array $\mathcal{O}$ combined by $\mathcal{O}_{net}$ and $\mathcal{O}_{flow}$ can directly act as the input state of E2eDRS. However, the scheduling performance is hard to reach our expectation mainly because of two reasons. First, the deterministic-related resource scheduling through network is time-correlated. Nevertheless, $\mathcal{O}$ only focuses on the scheduled resources and ignores the relationship between time sequence and deterministic-related resources. As a result, the temporal pattern of resource scheduling cannot be well-learned, which degrades the network performance. Second, the end-to-end resource scheduling relies on properly allocating the available resources to each flow along its path. But $\mathcal{O}$ only pays attention to the resource consumption on each link or node. It cannot well reflect the scheduled resources over the flow path. To overcome the above two deficiencies, we build an end-to-end resource consumption matrix, which can store the utilization of time-correlated resources along each flow path.

*Shining point 1: Build end-to-end resource consumption matrix.* Considering the deterministic-related resource scheduling is time-correlated, the end-to-end resource consumption matrix is assigned two dimensions. One dimension, i.e., each row of matrix, counts the time sequence of cycle time, where the cycle time for deterministic service is defined as the granularity of scheduling [33] in IEEE 802.1Q standards. The other dimension,

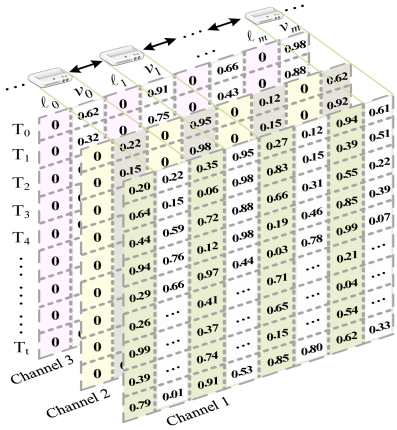


Fig. 2. Diagram of end-to-end resource consumption matrix.

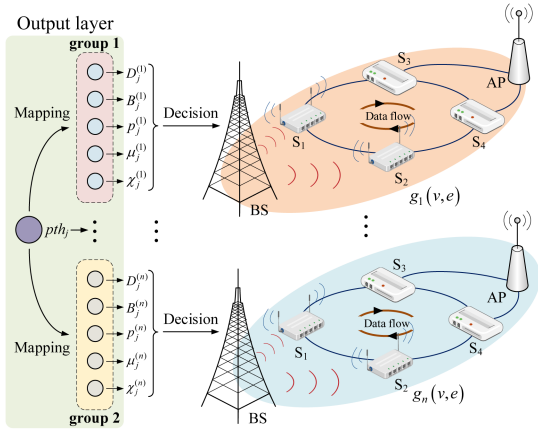


Fig. 3. Mechanism of action group selection and mapping policy between different networks $g_\epsilon(v,e)$ and the action groups.

i.e., each column of matrix, marks each link and node along the flow path. This matrix is embedded into the observation of E2eDRS algorithm. When a flow consumes the available resources at each cycle, the resource utilization along the flow path in the matrix can be updated, and the temporal pattern of resource scheduling can be well-learned. To observe each network state described by (16), multiple channels are introduced for the matrix. As shown in Fig. 2, the first channel includes the utilization of link capacity $\mathscr{H}_{\mathcal{C}_\ell}$ and CPU frequency $\mathscr{H}_{\mathcal{Q}_{v_i}}$ on each node. The second channel represents the energy utilization $\mathscr{H}_{E_{v_i}}$ on the node, and the other positions are padded with zero. The memory resource consumption $\mathscr{H}_{\mathcal{M}_{v_i}}$ of each node is stored by the third channel. This matrix can also assist E2eDRS extending into more generic time-related network resource scheduling scenarios, such as joint time-frequency domain scheduling, or other types of TDMA based resource scheduling in converged network.

The time slot constraints of (6) and (7) are very important to avoid the packet conflict during the same time. Note that the packet conflict can greatly decrease the number of schedulable flows. To satisfy with the time slot constraints, the agent needs to know which time slot on the link is valid for sending the coming flow under the current state. Thus, the shining point 2 is designed.

*Shining point 2: Set up collaboratively working mechanism between network states and actions.* To help the agent avoid packet conflict from different flows, choosing valid time slot for sending each flow is necessary. Therefore, an action array is embedded into the observation to collaboratively working with the network states. In the action array, the actions for choosing valid time slots are encoded by 1, while the invalid ones are set to zero. As such, the agent can identify the valid time slots and accommodate more flows.

To support both horizontal and vertical end-to-end deterministic-related network architectures, the deterministic-related resource scheduling in each $g_\epsilon(v,e)$ is needed. Therefore, the actions are grouped by $g_\epsilon(v,e)$, and the network with higher communication quality can be selected to transmit the flows. This technique is described by shining point 3.

*Shining point 3: Set up the action groups.* A vertical network consists of multiple horizontal networks with topology $g_\epsilon(v,e)$. Because a large number of deterministic-related resources are distributed into each $g_\epsilon(v,e)$, we need to figure out the relationship between each action and $g_\epsilon(v,e)$. As shown in Fig. 3, the neurons in output layer are chunked into multiple action groups. Each group maps a $g_\epsilon(v,e)$ in the vertical network. The horizontal end-to-end deterministic-related network architecture is a special case of vertical end-to-end deterministic-related network architecture such that $\epsilon = 1$. It means one action group is enough to indicate the horizontal end-to-end deterministic-related network architecture. Each action in different action groups are labeled by $a_i^{(\epsilon)} \in \mathcal{A}$, meaning that the resources are scheduled in $g_\epsilon(v,e)$. When the flow path is decided, the network $g_\epsilon(v,e)$ is actually chosen to transmit the flow. Then the corresponding action group takes effect, while the actions not belonging to this action group will be ignored.

To further improve the network scheduling ability, the risk-aware mechanism is introduced. This mechanism can assist the agent being aware of any risky actions decided by neural network at each step. The principle of risk-aware mechanism is demonstrated by shining point 4.

*Shining point 4: Risk-aware mechanism.* In risk-aware mechanism, each action decided by neural network is evaluated by 0–1 integer scores according to its impact on scheduling performance. Specifically, if the resource utilization limit is exceeded by taking the action decided at the current step, the agent will get score 0. To avoid scheduling failure, the agent needs to choose another action calculated by neural network until it gets score 1. The training loop will be finished when all the actions are marked by score 0. With this mechanism, the agent can spend fewer steps exploring better policies for improving the scheduling performance.

### B. Working Mechanism of E2eDRS Algorithm

The E2eDRS algorithm schedules the deterministic-related resources in horizontal and vertical end-to-end deterministic-related network architectures from end to end. Based on the pseudo codes of E2eDRS, the working mechanism can be summarized as follows.

*Initialization:* In line 1–4 of Algorithm 1, at the beginning of training process, the parameters of actor network and critic network are initialized by $\theta_0$ and $\upsilon_0$, respectively. In addition, the replay buffer and training correlated parameters are initiated. A counter is set to mark the current step, and avoid exceeding the maximal size of replay buffer.

*Mapping between state and action:* In line 5–16 of Algorithm 1, the input state $o_t$ embeds with the end-to-end resource consumption matrix and action array mentioned in shining point 1 and shining point 2, respectively. As such, the $o_t$ has four channels. The first three channels are used to schedule different deterministic-related resources from end to end. For determinacy, the transmission delay between any two hops are bounded within a cycle time. To observe the resource utilization on each cycle time during the end-to-end flow transmission, the number of rows in each channel should be equal to the hop count that the flow goes through. The column number of each channel depends on the amount of links and nodes in the network. The resources utilization in (16) are contained in the three channels of matrix. The last channel of $o_t$ maps the valid or invalid sending time slots under the current state.

Based on $o_t$, the agent makes an action decision according to the current optimal policy $a_t = \pi^*_{\theta_0}(o_t)$. The policy mapping from $\mathcal{O}$ to $\mathcal{A}$ follows the Gaussian distribution. Each value of resource scheduling decision in $a_t$ is then scaled into a valid range. Algorithm 2 describes the action decision process at each step. The input is original resource scheduling decisions grouped by $g_\epsilon(v,e)$, that is $a_t^0 = \{pth_t, \mathcal{D}_t^{(\epsilon)}, \mathcal{B}_t^{(\epsilon)}, p_t^{(\epsilon)}, \mu_t^{(\epsilon)}, \chi_t^{(\epsilon)}\}$, where $\epsilon = 1, 2, \ldots, n$. In essence, each flow path decision maps into the unique $g_\epsilon(v,e)$. Therefore, when the path is decided, only the resource decisions in the corresponding $g_\epsilon(v,e)$ can take effect. The key point is to find the mapping relationship between the decided $pth_t$ and $g_\epsilon(v,e)$, which is described by line 1–15 in Algorithm 2. First, all the flow paths with no loop are found for each $g_\epsilon(v,e)$. Then the network architecture set is built up, which consists of $n$ horizontal networks. By iteratively searching the network architecture set, the mapping from $pth_t$ to $g_h(v,e)$ can be found. The other optimal resources scheduling decisions in $g_h(v,e)$ are set to final decisions in line 16–17.

*State transition:* Before taking the resource scheduling decision $a_t$, the risk-aware mechanism is introduced to avoid exceeding upper limit of resource capacity. In line 7–10 of Algorithm 1, the agent evaluates the score of decided action $a_t$ by interacting with the environment offline. If the action causes failure of resource scheduling due to exceeding the network resource limit, the score is marked by 0, and the agent needs to evaluate the score of another action re-calculated by neural network with Gaussian noise based on the same input state. If the action $a_t$ can maintain the iteration loop of the algorithm during resource scheduling process, the score is marked by 1, and $a_t$ with score 1 is taken by the agent. If all the actions are marked by score 0, the agent performs $a_t$ with score 0 and the current iteration loop is finished. After the action is taken, an immediate reward can be obtained, and the state is transited to $o_{t+1}$.

**Algorithm 1.** DRL-Based End-to-End Deterministic-Related Resource Scheduling Algorithm.

**Input**: Initial parameters $\theta_0$ of actor network, initial parameter $\upsilon_0$ of critic network, buffer size $\Im$, trajectory length T.
**Output**: Optimal policy $\pi^* : \mathcal{O} \rightarrow \mathcal{A}$.

1 $\theta \leftarrow \theta_0$; $\upsilon \leftarrow \upsilon_0$; $count \leftarrow 0$;
2 **for** $iter \in \{0, 1, 2, \ldots, m\}$ **do**
3  Initialize the state $o$;
4  $done \leftarrow False$;
5  **while** *not done* **do**
6  Embed end-to-end resource consumption matrix and action array into $o_t$;
7  Generate deterministic-related resource scheduling actions by using Alg. 2;
8  Evaluate score for each action $a_t \in \mathcal{A}$ by interacting with environment offline;
9  Try to perform the action with $score$=1;
10  Perform the action $a_t$ decided at the beginning if all actions are marked by score 0;
11  Obtain an immediate reward $\zeta_{t+1}$ and transit state to $o_{t+1}$ by Alg. 3;
12  **if** *(Dissatisfy constraints Eq.* (3)*-Eq.* (11)*)|(All flows are scheduled)* **then**
13  $done \leftarrow True$;
14  **end**
15  Store tuple $(o_t, a_t, \zeta_{t+1}, o_{t+1}, done)$ into the $count$-th line of replay buffer;
16  $count \leftarrow count + 1$;
17  **if** $count == \Im$ **then**
18  $count \leftarrow 0$;
19  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ from replay buffer;
20  **for** *each trajectory* $\tau \in \mathcal{D}_k$ **do**
21  Define $\delta_t \leftarrow \zeta_{t+1} + \omega V_{\upsilon_0}(o_{t+1}) - V_{\upsilon_0}(o_t)$;
22  Compute advantage estimates $\hat{A}_t$ for each $o_t \in \tau$ started by step $t_0$: $\hat{A}_t \leftarrow \sum_{l=0}^{t_0+T-1-t} (\lambda\omega)^l \delta_{t+l}$;
23  Update $\theta$ of actor network by maximizing the objective: $J(\theta) = \frac{1}{T}\sum_{t=t_0}^{t_0+T-1} min\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}\hat{A}_t, clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}_t\right]$, via stochastic gradient ascent step;
24  Update $\upsilon$ of critic network by minimizing the loss: $L(\upsilon) = \frac{1}{T}\sum_{t=t_0}^{t_0+T-1}(V_{\upsilon_0}(o_t) + \hat{A}_t - V_\upsilon(o_t))^2$, via stochastic gradient descent step;
25  **end**
26  $\theta_0 \leftarrow \theta$; $\upsilon_0 \leftarrow \upsilon$;
27  **end**
28  **if** $done$ **then**
29  break;
30  **end**
31  **end**
32 **end**

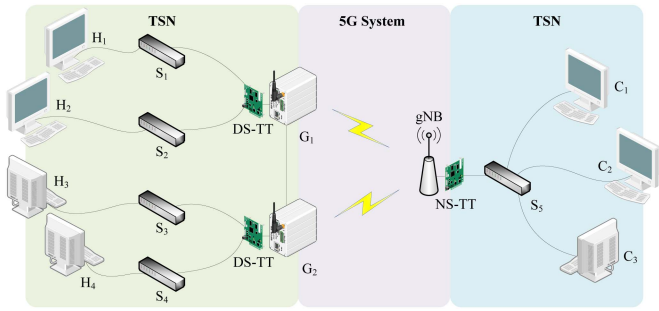


Fig. 4. Horizontal scheduling scenario in a TSN-5G network.

The state transition is described by Algorithm 3 in detail. According to the sending time of each flow and time sequence along the flow path, the end-to-end resource utilization can be filled into the state matrix. Specifically, the sending time and slot length decisions are related to time slot resources. Therefore, we have $\mathcal{D}_t = \{\mathcal{D}_t^{\mathcal{T}}, \mathcal{D}_t^i\}$, where $\mathcal{D}_t^{\mathcal{T}}$ and $\mathcal{D}_t^i$ are sending time slot and slot length for flow transmission on each hop, respectively. Due to the bounded latency of passing through each hop, we use the link-node pairs to indicate a full flow path, that is $pth_t = \{(\ell_j, v_j)\}$, where $j = 1, 2, \ldots,$ m. When the flow goes through a link-node pair, the sequence number of $\mathcal{D}_t^{\mathcal{T}}$ plus 1. On each link-node pair, the state of four channels are updated. Let $S_c(\mathcal{D}_t^i)$ and $S_p(\mathcal{D}_t^i)$ be the link capacity and CPU frequency resource utilization within time slot $\mathcal{D}_t^i$, respectively. Line 3-5 of Algorithm 3 updates the total link capacity and CPU frequency usage on line $\mathcal{D}_t^{\mathcal{T}}$. By calculating the current utilization of computing energy resource $S_e(\mathcal{D}_t^i)$, line 6–7 updates the total computing energy usage on each node. According to the memory utilization $S_m(\mathcal{D}_t^i)$ at current step, line 8–9 updates the total memory usage on each node. Line 10–15 updates the action array, where the valid sending time slots for the coming flow are encoded by 1. As a result, the next state $o_{t+1}$ is composed by the four channels, that is $o_{t+1} = \{o_{t+1}^1, o_{t+1}^2, o_{t+1}^3, o_{t+1}^4\}$ in line 19.

Furthermore, line 15–19 of Algorithm 1 checks the shutdown criteria, and stores the state transition tuple into the replay buffer. The counter records current step count.

*Training approach:* Line 17–32 of Algorithm 1 shows the training process of E2eDRS algorithm. The agent collects set of trajectories from the replay buffer. Each trajectory length is set to T, where $T < \Im$. For a trajectory that starts with step $t_0$, E2eDRS updates the actor network by optimizing the policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$ with the following function.

$$J(\theta) = \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \min \left[ r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right], \tag{22}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$ represents the gap between old and new policies. The advantage function $\hat{A}_t$ guides the direction of policy update. The $\hat{A}_t$ is defined as

$$\hat{A}_t = \sum_{l=0}^{t_0+T-1-t} (\lambda\omega)^l \delta_{t+l}, \tag{23}$$

**Algorithm 2:** Generation of Actions at Each Step.

**Input**: Original action
$a_t^0 = \{pth_t, \mathcal{D}_t^{(\epsilon)}, \mathcal{B}_t^{(\epsilon)}, p_t^{(\epsilon)}, \mu_t^{(\epsilon)}, \chi_t^{(\epsilon)}\}$
determined by actor network according to current optimal policy $a_t^0 = \pi_{\theta_0}^*(o_t) + \mathcal{N}$.
**Output**: Choosed action $a_t$.

```
1  find all paths {pth_1, pth_2, ..., pth_k} with no loop for each g_ε(v, e);
2  Create network architecture set N = {g_1(v,e), g_2(v,e), ..., g_n(v,e)};
3  dw ← 0; h ← 0;
4  for i ∈ {0, 1, 2, ..., n − 1} do
5      for j ∈ {0, 1, 2, ..., N[i].length − 1} do
6          if pth_t == N[i][j] then
7              h ← i + 1;
8              dw ← 1;
9              break;
10         end
11     end
12     if dw then
13         break;
14     end
15 end
16 Choose the action from a_t^0 that belongs to g_h(v, e):
   D_t ← D_t^(h); B_t ← B_t^(h); p_t ← p_t^(h); μ_t ← μ_t^(h);
   χ_t ← χ_t^(h);
17 a_t ← {pth_t, D_t, B_t, p_t, μ_t, χ_t};
```

where $\delta_t = \zeta_{t+1} + \omega V_{\upsilon_0}(o_{t+1}) - V_{\upsilon_0}(o_t)$. When $\hat{A}_t > 0$, it encourages the current policy update by increasing the probability $\pi_\theta(a_t|s_t)$. When $\hat{A}_t < 0$, it reduces the probability $\pi_\theta(a_t|s_t)$. To prevent the excessive update of policy, the clip mechanism is introduced to limit the policy gap $r_t(\theta)$ into $[1-\epsilon, 1+\epsilon]$. To update the parameters in critic network, we compute the difference between the target and evaluate state values as the loss function, which is

$$L(\upsilon) = \frac{1}{T} \sum_{t=t_0}^{t_0+T-1} \left( V_{\upsilon_0}(o_t) + \hat{A}_t - V_\upsilon(o_t) \right)^2. \tag{24}$$

After updating the neural networks, the old parameters $\theta_0$ and $\upsilon_0$ are replaced by the new ones $\theta$ and $\upsilon$.

## V. Performance Evaluation

This section lists the simulation settings of horizontal and vertical scheduling scenarios, and the performance of E2eDRS is evaluated in different network architectures. The training environment is built on Ubuntu 20.04 (kernel 5.11.0-41-generic) system with Python implementation. We implement the algorithms on the central controller with high-performance hardwares CPU (AMD R9 5950X), GPU (NVIDIA QUADRO RTX 6000), and 64GB Kingston RAM.

### A. Horizontal Scheduling Scenario

*Settings for horizontal end-to-end deterministic-related network architecture:* We use the application-oriented

**Algorithm 3:** State Transition Approach.

**Input**: Current state $o_t = \{o_t^1, o_t^2, o_t^3, o_t^4\}$, action $a_t = \{pth_t, \mathcal{D}_t, \mathcal{B}_t, p_t, \mu_t, \chi_t\}$, where $pth_t = \{(\ell_j, v_j)\}$, $\mathcal{D}_t = \{\mathcal{D}_t^{\mathcal{T}}, \mathcal{D}_t^i\}$.
**Output**: Next state $o_{t+1}$.

**for** $i \in \{0, 1, 2, \ldots, pth_t.length - 1\}$ **do**
  **for** $j \in \{0, 1, 2, \ldots, pth_t.length - 1\}$ **do**
    Update channel 1 by:
    $o_{t+1}^1[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].\ell] \leftarrow o_t^1[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].\ell] + S_c(\mathcal{D}_t^i)$;
    $o_{t+1}^1[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] \leftarrow o_t^1[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] + S_p(\mathcal{D}_t^i)$;
    Update channel 2 by:
    $o_{t+1}^2[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] \leftarrow o_t^2[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] + S_e(\mathcal{D}_t^i)$;
    Update channel 3 by:
    $o_{t+1}^3[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] \leftarrow o_t^3[\mathcal{D}_t^{\mathcal{T}}][pth_t[j].v] + S_m(\mathcal{D}_t^i)$;
    Update channel 4 by:
    **if** $check\_valid(\mathcal{D}_t^{\mathcal{T}}, o_t)$ **then**
      $o_{t+1}^4[\mathcal{D}_t^{\mathcal{T}}, :] \leftarrow 1$;
    **else**
      $o_{t+1}^4[\mathcal{D}_t^{\mathcal{T}}, :] \leftarrow 0$;
    **end**
  **end**
  $\mathcal{D}_t^{\mathcal{T}} \leftarrow \mathcal{D}_t^{\mathcal{T}} + 1$;
**end**
Assign the channels to $o_{t+1}$ by:
$o_{t+1} \leftarrow \{o_{t+1}^1, o_{t+1}^2, o_{t+1}^3, o_{t+1}^4\}$;

TSN-5G network to simulate the horizontal end-to-end deterministic-related network architecture. The benchmark topology is shown in Fig. 4. The TSN hosts are accessed through wired TSN. According to 3rd Generation Partnership Project (3GPP), 5GS is configured as a logical TSN switch to seamlessly transmit data forwarded by the upstream TSN switches [34]. Device-side (DS-TT) and network-side (NS-TT) TSN translators in 5G network are used to control the sending time of TT flows and achieve the deterministic transmission. We use the CQF mechanism to guarantee the deterministic flow transmission over the TSN-5G network. The link bandwidth of wired TSN is 1000 Mbps. The CPU frequencies of servers are 4.2 GHz, 2.4 GHz, and 10 GHz, respectively. 1,000 TT flows are generated from TSN hosts $H_1 \sim H_4$, and they are sent to high-performance servers $C_1 \sim C_3$. The flow size is set to 50B∼1 KB [35], and the flow periods include {4 ms, 8 ms, 16 ms}. The transmitted TT flows have different cpu frequency requirements that obey the normal distribution with mean value of 16 MHz and variance of 3. The cycle time should be smaller than maximum common divisor of flow periods. Hence, the cycle time is set to {1 ms, 2 ms, 4 ms}.

In 5G network, 14 OFDM symbols are included during a transmission time interval (TTI) with 15kHz sub-carrier spacing, which means the TTI is 1 ms. We use 3.7∼3.8 GHz frequency band, and the system bandwidth is 40 MHz. The power of 5G base station (BS) is set to 40 dBm, while the transmitting power of gateway is 20 dBm. The 5G channel gain is described by indoor hotspot LOS scenario [36], which is $PL(dB) = 32.4 + 17.3\log_{10}(l_{5G}) + 20\log_{10}(f_c)$, where we set $l_{5G} = 100$ m denoting the distance between sensors and BS. In addition, the Gaussian noise power is $-104$ dBm. Table I lists the network parameters in detail.

TABLE I
TSN-5G NETWORK PARAMETERS

| 5G Network | | Wired TSN | |
|---|---|---|---|
| 5G bandwidth | 40 MHz | Link bandwidth | 1000 Mbps |
| Frequency band | 3.7∼3.8 GHz | Cycle time | {1,2,4}ms |
| 5G TTI | 1 ms | Packet size | 50 B∼1 KB |
| BS power | 40 dBm | Flow periods | {4,8,16}ms |
| Gateway power | 20 dBm | $C_1$ frequency | 4.2 GHz |
| 5G distance | 100 m | $C_2$ frequency | 2.4 GHz |
| Noise power | -104 dBm | $C_3$ frequency | 10 GHz |

*Formulation of optimization problem:* Based on (12), our optimization goals are to optimize the schedulable number of flows and load balance of computing resource utilization. Our designed reward function can intuitively reflect the quality of optimization performance. By substituting the flow number statistic function of (20) into reward function of (19), the reward $\zeta_{t+1}^1$ of scheduled flows can be obtained. Similarly, reward $\zeta_{t+1}^2$ of load balance can be acquired by substituting the load balance statistic function of (21) into (19). To improve the scheduling performance, the rewards need to be optimized. Hence, the optimization goals can be achieved by

$$Maximize \quad a \times \zeta_{t+1}^1 + b \times \zeta_{t+1}^2, \tag{25}$$

where $a, b$ are weights of each objective function. Besides, the deterministic transmission constraints need to satisfy with (3)-(11). Thus, it is subject to

$$\begin{cases} \text{Deterministic transmssion constraints (3)–(5);} \\ \text{Network resource constraints (6)– (11).} \end{cases} \tag{26}$$

*Settings for E2eDRS algorithm:* To extract the features of input matrix, E2eDRS uses two convolution layers, each layer has $3 \times 3$ kernel size. Due to the horizontal end-to-end deterministic-related network architecture, the action group is set to 1. The training episode is set to 1e6. In each episode, the agent needs to experience a complete state sequence. Based on our experience, the replay buffer size is 2048 to accommodate enough trajectories. The sampled trajectory length is set to 64. To perceive the long-term reward, the discounting factors $\lambda$ and $\omega$ are set to 0.99 and 0.95, respectively. We use the Adam optimizer to update the actor and critic networks, and the learning rate $lr_a$ and $lr_c$ for actor and critic networks are both 3e-4. The detailed parameter description is shown in Table II.

We compare E2eDRS with the traditional non end-to-end DRL-based scheduling algorithm, namely multi-private deterministic-related resource scheduling (MultiDRS) approach. MultiDRS implements the private algorithm for each part of network in TSN-5G system. Specifically, MultiDRS uses state-of-the-art algorithm, TimeDRS, reproduced from [17] for

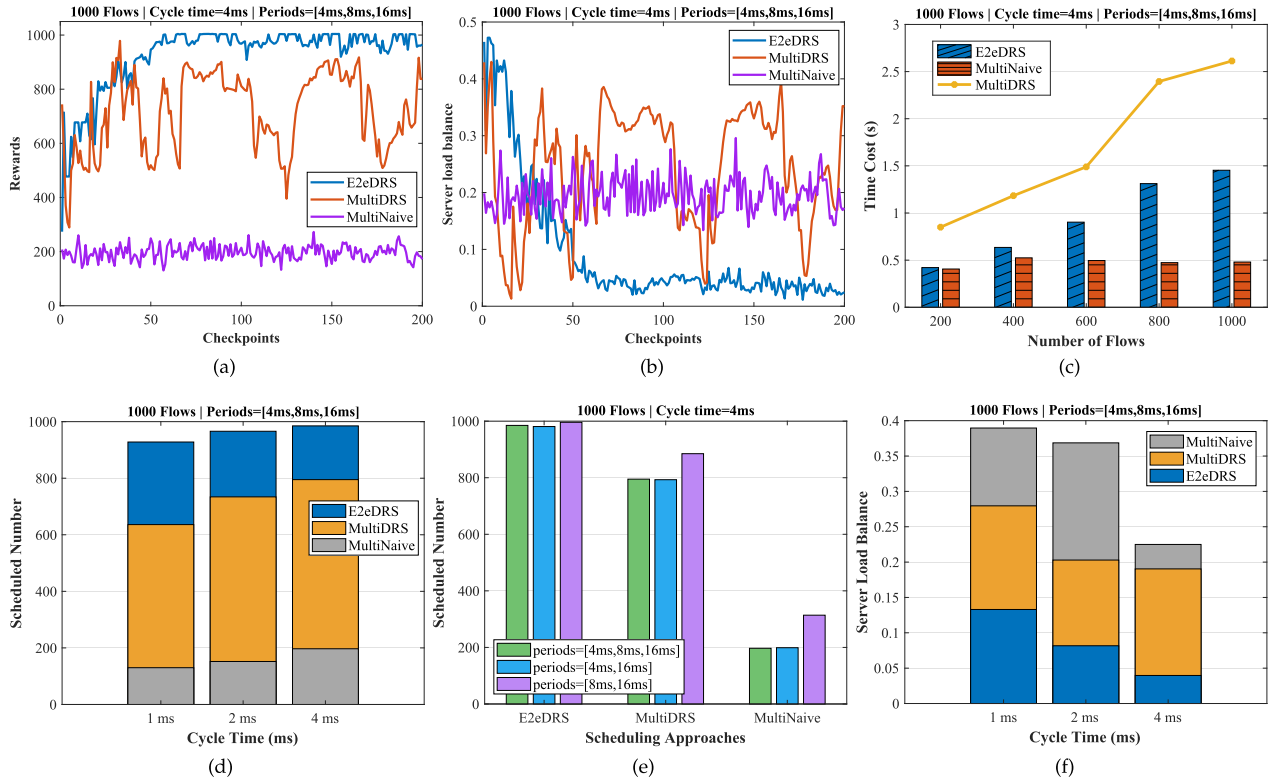


Fig. 5. Horizontal scheduling results. (a) Rewards on each checkpoint; (b) Load balance on servers; (c) Computational time cost of various resource scheduling algorithms; (d) Scheduled flow number with different cycle time; (e) Scheduled flow number with different flow periods; and (f) Server load balance with different cycle time.

TABLE II
PARAMETERS FOR LEARNING ALGORITHMS

| Learning para. | Value | Learning para. | Value |
|---|---|---|---|
| Optimizer | Adam | Buffer size | 2048 |
| Episode number | 1e6 | Trajectory length | 64 |
| Kernel size | 3×3 | Coefficient a | -1 |
| $D_k$ length | 10 | Coefficient b | -0.8 |
| Discounting factor $\lambda$ | 0.99 | Learning rate $lr_a$ | 3e-4 |
| Discounting factor $\omega$ | 0.95 | Learning rate $lr_c$ | 3e-4 |

wired TSN network, and the K-DDPG resource scheduling algorithm [37] for 5G network. The training parameters of MultiDRS are the same with E2eDRS.

The scheduling results and analysis of horizontal end-to-end deterministic-related network architecture are described as follows.

*Training effectiveness:* The training processes of different algorithms are shown in Fig. 5(a) and (b). For every 5e3 training episodes, we set a checkpoint. Fig. 5(a) depicts the total reward on each check point. E2eDRS obtains the highest reward compared with the MultiDRS and MultiNaive scheduling approaches. In addition, the reward variance of E2eDRS is much more smaller than MultiDRS after reaching the convergence. This is because the global end-to-end deterministic scheduling of E2eDRS can collaboratively allocate various resources across heterogeneous networks. Different algorithms in MultiDRS only pursue the maximal reward in their own networks, none of an algorithm takes care of the global network resources utilization. Even if it reaches the optimal in a single network, it is still difficult to achieve the overall optimal from end to end. The MultiNaive algorithm randomly schedules the deterministic-related resources in each part of network. Although the computational time is short for MultiNaive, its obtained rewards steadily maintain at lower values.

Fig. 5(b) shows the training process of computing resource load balance on each high-performance server. E2eDRS hits the smallest standard deviation on load balance, which means the training performance of E2eDRS is the best among them. However, the variance of MultiDRS is too large to meet our expectations. Even the average reward of MultiDRS is close to MultiNaive. As a result, MultiDRS is hard to achieve the trade-off between schedulable flow number and server load balance.

*Computational time cost:* The computational time cost of different scheduling approaches are depicted by Fig. 5(c). With the growing number of added flows, the computational time cost of E2eDRS and MultiDRS are getting larger. Because the algorithm needs to spend more time steps to schedule these flows. In addition, MultiDRS consists of multiple private algorithms for specific network, and it has more number of neural networks to compute. As a result, MultiDRS needs more time to make the inference. The computational time cost of MultiNaive steadily maintains at low value. This is because the scheduling ability is too limited for MultiNaive. No matter how many flows are added into the network, MultiNaive can always schedule 200 flows on average. Thus, the running steps are almost the same at each time for scheduling.

*Flow scheduling ability:* To evaluate the scalability of E2eDRS, we change the network parameters, including cycle time and flow period to observe the scheduling performance of different approaches. In Fig. 5(d), E2eDRS can always schedule the maximal number of TT flows with different cycle time. The schedulable flow number of E2eDRS averagely increases by 1.33x and 6.01x than MultiDRS and MultiNaive, respectively. Additionally, the cycle time can also impact the schedulable number of flows. Fig. 5(d) shows that each scheduling approach can schedule more flows when the cycle time gets larger. For example, the number of flows scheduled by MultiNaive, MultiDRS, and E2eDRS in 4 ms cycle time increases by 51.54%, 25%, and 6.14%, compared with the flows scheduled during 1 ms cycle time, respectively. Two main reasons can lead to this result. First, larger cycle time enlarges its time slot capacity to accommodate more flows. Second, the number of cycles decreases in a hyper period with the cycle getting large. The smaller number of cycles can reduce the searching difficulty of scheduling algorithms. Therefore, the scheduling algorithms can schedule more flows with a large cycle time.

*Impact of flow periods:* We evaluate the impact of different flow periods in Fig. 5(e). Experimental results show that the scheduled flow numbers are almost the same between flow period {4 ms, 8 ms, 16 ms} and {4 ms, 16 ms}, while the number of schedulable flows is larger under the flow period of {8 ms, 16 ms}. For MultiDRS and MultiNaive, the schedulable flow number with period {8 ms, 16 ms} increases by 11.32% and 59.39%, respectively compared with flow period {4 ms, 8 ms, 16 ms}. It demonstrates that the flows with shorter period have more impact on scheduling ability because these flows appear more frequently within a hyper period and they can occupy more time slot resources. Moreover, E2eDRS can also schedule the most number of flows with different flow periods. Fig. 5(e) shows that the schedulable flow number of E2eDRS with various flow periods grows by 1.19x and 4.17x on average compared with MultiDRS and MultiNaive, respectively.

*Server load balance:* The scheduling performance on server load balance is evaluated in Fig. 5(e). The server load balance is measured by the standard deviation of computing resource utilization on each server based on (21). The E2eDRS can averagely optimize 2.65x and 3.87x value of server load balance than MultiDRS and MultiNaive, respectively. It illustrates that E2eDRS is good at learning the optimal resource allocation policy from a holistic perspective. Furthermore, each scheduling approach performs well with the large cycle time. This is because less number of cycles within a hyper period can reduce the difficulty for scheduling approaches to search the optimal policy. Especially for E2eDRS, it can reduce the standard deviation of load balance by 62.79% by changing the cycle time from 1 ms to 2 ms. The standard deviation with 4 ms cycle time is 2.06x lower than that with 2 ms.

## B. Vertical Scheduling Scenario

*Settings for vertical scheduling:* The vertical scheduling scenario is to schedule the deterministic-related resources in vertical end-to-end deterministic-related network architecture,

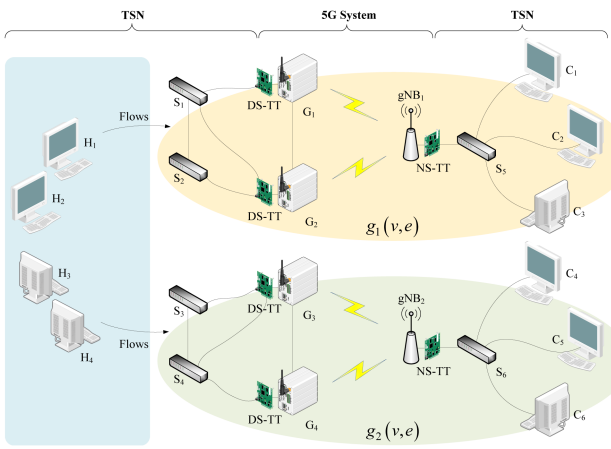


Fig. 6. Vertical scheduling scenario consists of two paralleled horizontal end-to-end deterministic-related network architectures. During the flow transmission process, the network with higher communication quality in the architecture can be selected to transmit the flows.

which includes two paralleled horizontal networks $g_1(v,e)$ and $g_2(v,e)$ shown in Fig. 6. Hosts $H_1 \sim H_4$ can select a better horizontal network to send the flows. Each horizontal network consists of TSN-5G system. To maintain the total computing resources, the CPU frequencies of each server $C_1 \sim C_6$ are set to 2.4 GHz, 2.5 GHz, 1.4 GHz, 4.5 GHz, 4.8 GHz, and 1 GHz, respectively. Due to the vertical scheduling with $n = 2$, the action group of E2eDRS algorithm should be 2. The other network parameters are the same with Tables I and II. The objective goals of vertical scheduling are still to optimize the schedulable number of flows and server load balance. Thus, the formulated optimization problem of this scheduling scenario is described by (25) and (26).

The experimental results show that our proposed scheduling approach can be applied into the vertical end-to-end deterministic-related network architecture, and the scheduling performance is evaluated as follows.

*Training effectiveness:* The training processes of total rewards and server balance in vertical scheduling scenario are depicted by Fig. 7(a) and (b). After convergence, Fig. 7(a) shows that E2eDRS can improve the number of schedulable flows by 1.33x and 14.64x, respectively compared with MultiDRS and MultiNaive algorithms. With the advantage of globally end-to-end resource scheduling, E2eDRS has smaller reward variance than MultiDRS during training. However, the MultiNaive algorithm underperforms in vertical scheduling scenario. Compared with the horizontal scheduling scenario, the vertical scheduling of MultiNaive averagely reduces the reward value by 2.98x, which means the MultiNaive cannot well utilize the deterministic-related resources distributed in each paralleled horizontal network.

Fig. 7(b) shows that E2eDRS can reduce 2.43x server load balance than MultiDRS. Because fewer flows can be successfully scheduled by MultiNaive, each server is under the light loads. As such, the server load balance of MultiNaive is better than MultiDRS. Fortunately, E2eDRS can achieve the same load balance level with MultiNaive. This proves E2eDRS can make

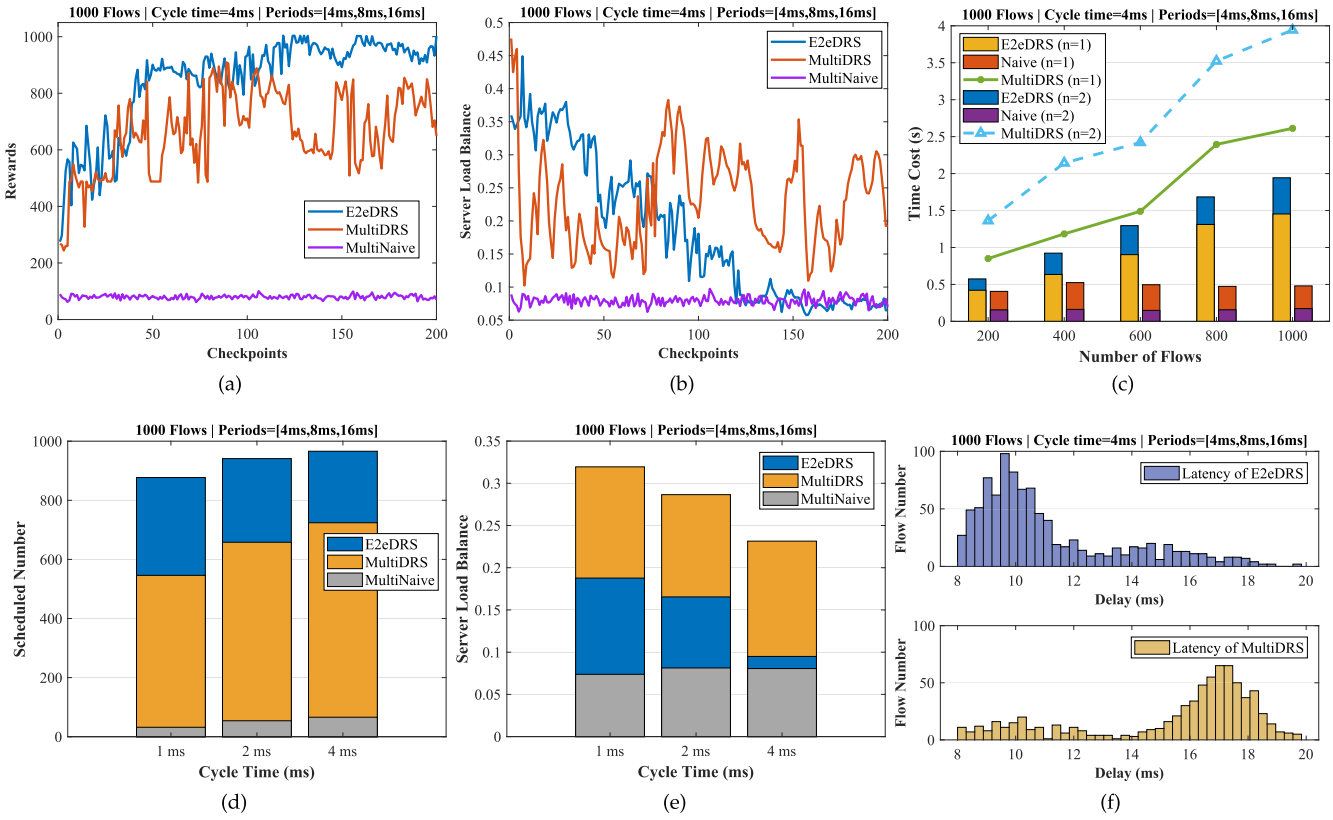


Fig. 7. Vertical scheduling results. (a) Rewards on each checkpoint; (b) Server load balance on each checkpoint; (c) Computational time cost comparison of different algorithms; (d) Scheduled flow number with different cycle time; (e) Server load balance with different cycle time; and (f) End-to-end latency comparison of different algorithms.

a good trade-off between schedulable flow number and server load balance.

*Computational time cost:* Fig. 7(c) compares the computational time cost of different algorithms in horizontal ($n = 1$) and vertical networks ($n = 2$), respectively. Because the scheduled flow number can affect the computational time cost, E2eDRS and MultiDRS consume more time to compute the scheduling results with the increasing number of added flows. Moreover, the scale of network topology can impact the computational time cost. For E2eDRS and MultiDRS, the computational time cost with $n = 2$ is larger than that with $n = 1$ because the number of nodes is larger with $n = 2$. On the other hand, the MultiDRS has more private scheduling approaches to deal with each part of network in vertical end-to-end deterministic-related network architecture. Hence, MultiDRS has larger computational time cost in vertical end-to-end deterministic-related network architecture. In contrast, the time cost of MultiNaive with $n = 2$ is shorter than that with $n = 1$. It is because the scheduled flow number of MultiNaive in vertical scheduling scenario is much smaller than horizontal scheduling, and the flow number has more effect on computational time of MultiNaive algorithm.

*Flow scheduling ability:* As shown in Fig. 7(d), the E2eDRS can schedule the most number of TT flows. Compared with MultiDRS and MultiNaive, the E2eDRS can averagely increase the schedulable flow number by 1.44x and 18.2x, respectively in the vertical scheduling scenario. Similarly to the horizontal scheduling, the larger cycle time can accommodate more flows in vertical network.

Fig. 7(e) shows the server load balance scheduled by various algorithms. Different from the scheduling results in horizontal end-to-end deterministic-related network architecture, MultiNaive achieves the best server load balance for vertical scheduling. This is because the MultiNaive schedules fewer number of TT flows in vertical network so that the computing resource utilization on each server is too small. And the overall value of (21) can also be small.

*End-to-end latency comparison:* The end-to-end transmission latency of TT flows scheduled by E2eDRS and MultiDRS algorithms are depicted in Fig. 7(f). The E2eDRS algorithm focuses on the resource allocation from end to end. Hence, E2eDRS can globally plan the flow sending time from source host, and then it transmits the flows in high communication quality network with short paths. As such, the end-to-end latency scheduled by E2eDRS can be relatively small. In contrast, the MultiDRS algorithm focuses on the scheduling performance of each distributed network. To improve the schedulable flow number in each network, MultiDRS tends to route each flow to the path with sparse traffic. Thus, the end-to-end latency of MultiDRS is almost 73.16% larger than E2eDRS.

## VI. Conclusion

In this paper, we have investigated the end-to-end deterministic guarantee problem across the whole converged network. To manage the end-to-end flow transmission both in horizontal and vertical end-to-end deterministic-related network architectures, we have set up the global end-to-end control plane. By abstracting the deterministic-related network resources, the end-to-end deterministic-related resource abstraction model has been established in control plane to guarantee the determinacy across the converged network. The resource abstraction can assist the model working well for different underlying technologies. To fully utilize the network resources in the model, the E2eDRS algorithm has been proposed to jointly schedule the resources for each flow from end to end. Simulation results have demonstrated that our proposed method can achieve a good trade-off between computational time cost and schedulability for both horizontal and vertical scheduling scenario. Extended from the existing deterministic technologies that focus on time slot and frequency resources, our end-to-end deterministic-related resource abstraction model jointly considers more deterministic-related resources, such as routing, power, computing, memory, and energy resources. Different from the existing DRL-based scheduling algorithms only scheduling the resources in their isolated and fixed networks, E2eDRS can focus on end-to-end deterministic-related resource scheduling over the converged network. In addition, E2eDRS can also support varying network dimensions both in horizontal and vertical end-to-end deterministic-related network architectures. For the future work, we will investigate how to configure the data plane for converged network to achieve end-to-end determinacy by considering more deterministic-related resources.

## References


[1] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Optimization of flow allocation in asynchronous deterministic 5G transport networks by leveraging data analytics," *IEEE Trans. Mob. Comput.*, vol. 22, no. 3, pp. 1672–1687, Mar. 2023.

[2] Y. Zhang, Q. Xu, M. Li, C. Chen, and X. Guan, "QoS-aware mapping and scheduling for virtual network functions in industrial 5G-TSN network," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[3] W. Quan, J. Yan, X. Jiang, and Z. Sun, "On-line traffic scheduling optimization in IEEE 802.1Qch based time-sensitive networks," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.*, 2020, pp. 369–376.

[4] D. Yang et al., "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5162–5178, May 2024.

[5] G. Patti, L. L. Bello, and L. Leonardi, "Deadline-aware online scheduling of TSN flows for automotive applications," *IEEE Trans. Ind. Inf.*, vol. 19, no. 4, pp. 5774–5784, Apr. 2023.

[6] D. Ginthör, J. von Hoyningen-Huene, R. Guillaume, and H. Schotten, "Analysis of multi-user scheduling in a TSN-enabled 5G system for industrial applications," in *Proc. IEEE Int. Conf. Ind. Internet*, 2019, pp. 190–199.

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

[8] B. Wu, S. Wang, J. Wang, W. Tan, and Y. Liu, "Flexible design on deterministic IP networking for mixed traffic transmission," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 4360–4365.

[9] D. Ginthör, R. Guillaume, M. Schüngel, and H. D. Schotten, "Robust end-to-end schedules for wireless time-sensitive networks under correlated large-scale fading," in *Proc. 17th IEEE Int. Conf. Factory Commun. Syst.*, 2021, pp. 115–122.

[10] 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./Apr. 2022.

[11] M. Li, S. Guo, C. Chen, C. Chen, X. Liao, and X. Guan, "DecAge: Decentralized flow scheduling for industrial 5G and TSN integrated networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 1, pp. 543–555, Jan./Feb. 2024.

[12] X. Jiang et al., "Spatio-temporal routing, redundant coding and multipath scheduling for deterministic satellite network transmission," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 2860–2875, May 2023.

[13] 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. Inf.*, vol. 18, no. 10, pp. 7351–7360, Oct. 2022.

[14] G. Peng, S. Wang, Y. Huang, T. Huang, and Y. Liu, "Enabling deterministic tasks with multi-access edge computing in 5G networks," *IEEE Commun. Mag.*, vol. 60, no. 8, pp. 36–42, Aug. 2022.

[15] F. Song, L. Li, I. You, and H. Zhang, "Enabling heterogeneous deterministic networks with smart collaborative theory," *IEEE Netw.*, vol. 35, no. 3, pp. 64–71, May/Jun. 2021.

[16] H. Zhang, B. Feng, and A. Tian, "A systematic review for smart identifier networking," *Sci. China Inf. Sci.*, vol. 65, no. 12, Oct. 2022, Art. no. 221301.

[17] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, "Burst-aware time-triggered flow scheduling with enhanced multi-CQF in time-sensitive networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2809–2824, Dec. 2023.

[18] Huawei Technologies Co., Ltd., "5GDN industry white paper," 2019. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/news/5gdn-industry-white-paper-en.pdf

[19] M. C. Lucas-Estañ and J. Gozalvez, "Sensing-based grant-free scheduling for ultra reliable low latency and deterministic beyond 5G networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4171–4183, Apr. 2022.

[20] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Trans. Mob. Comput.*, vol. 20, no. 1, pp. 19–31, Jan. 2021.

[21] 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.

[22] 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.*, 2022, pp. 1–6.

[23] 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.

[24] Z. Li et al., "An enhanced reconfiguration for deterministic transmission in time-triggered networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1124–1137, Jun. 2019.

[25] 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.

[26] K. Gong, D. Yang, W. Zhang, and J. Ren, "An efficient scheduling approach for multi-level industrial chain flows in time-sensitive networking," *Comput. Netw.*, vol. 221, Feb. 2023, Art. no. 109516.

[27] H. Jiang, Z. Xiao, Z. Li, J. Xu, F. Zeng, and D. Wang, "An energy-efficient framework for Internet of Things underlaying heterogeneous small cell networks," *IEEE Trans. Mob. Comput.*, vol. 21, no. 1, pp. 31–43, Jan. 2022.

[28] W. Wu et al., "AI-native network slicing for 6G networks," *IEEE Wirel. Commun.*, vol. 29, no. 1, pp. 96–103, Feb. 2022.

[29] Y. Shi, S. Xia, Y. Zhou, Y. Mao, C. Jiang, and M. Tao, "Vertical federated learning over cloud-RAN: Convergence analysis and system optimization," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 2, pp. 1327–1342, Feb. 2024.

[30] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[31] W. Wu et al., "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, Jan. 2021.

[32] W. Wu et al., "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, Apr. 2023.

[33] IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment 29: Cyclic Queuing and Forwarding, IEEE Std. 802.1Qch-2017, Jun. 2017, pp. 1–30.
[34] 3GPP, "TS 23.501 V16.7.0 system architecture for the 5G system," Jan. 2021. https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.07.00_60/ts_123501v160700p.pdf
[35] Industrial Internet Consortium, "Time sensitive networks for flexible manufacturing testbed characterization and mapping of converged traffic types," Mar. 2019. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_TSN_Testbed_Char_Mapping_of_Converged_Traffic_Types_Whitepaper_20180328.pdf
[36] M. Series, "Guidelines for evaluation of radio interface technologies for IMT-2020," Tech. Rep. ITU, 2512.0, Oct. 2017.
[37] Z. Gu et al., "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2014–2028, Jul. 2021.

**Zongrong Cheng** (Member, IEEE) received the PhD degree in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2024. He is currently an engineer working in Shanghai Aerospace Electronic Technology Institute, Shanghai, China. His research interests include industrial Internet of Things, deterministic networks, resource management, data analytics, and deep reinforcement learning.

**Weiting Zhang** (Member, IEEE) received the PhD degree in communication and information systems with the Beijing Jiaotong University, Beijing, China, in 2021. From 2019 to 2020, he was a visiting PhD student with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Starting from 2021, he works as an associate professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include industrial Internet of Things, deterministic networks, edge intelligence, and machine learning for network optimization.

**Dong Yang** (Member, IEEE) received the PhD degree in communication engineering from the School of Electronics and Information Engineering, Beijing Jiaotong University, China, in 2008. From 2009 to 2010, he worked as a postdoctoral researcher in 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 of communication engineering and the director with Intelligent Industry Network Research Institute, Beijing Jiaotong University. His current research interests focus on network architecture, Internet of Things, wireless sensor networks and data driven network resource management. He has authored or coauthored more than 70 scientific publications in the field of network and communication technologies. More than 20 of his invention patents were licensed to ZTE. He has been serving as the associate editor for *IEEE Transactions on Mobile Computing* and the executive editor for *Transactions on Emerging Telecommunications Technologies* (Wiley). He has served as the guest editor for *IEEE Transaction on Industrial Informatics* in 2021. He has served as co-chair of IEEE INFOCOM 2023 Workshop PerAI-6 G, Session Chair of IEEE ICCT 2022, TPC member of IEEE MetaCom 2023, etc.

**Chuan Huang** (Member, IEEE) received the PhD degree in electrical engineering from Texas A&M University, College Station, USA, in 2012. From 2012 to 2014, he was a research associate and then a research assistant professor with Princeton University and Arizona State University, Tempe, respectively. He is currently an associate professor with the Chinese University of Hong Kong, Shenzhen. His current research interests include wireless communications and signal processing. He served as a Symposium Chair for IEEE GLOBECOM 2019 and IEEE ICCC 2019 and 2020. He has been serving as an editor for *IEEE Transactions on Wireless Communications*, *IEEE ACCESS*, *Journal of Communications and Information Networks*, and *IEEE Wireless Communications Letters*.

**Hongke Zhang** (Fellow, IEEE) received the MS and PhD 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 postdoctoral 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 Lab on Next Generation Internet Technologies. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He is the author of more than ten books and the holder of more than 70 patents. He is the chief scientist with the National Basic Research Program of China (973 Program) and has also served on the editorial boards of several international journals.

**Xuemin Sherman Shen** (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, 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 in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the technical program committee chair/co-chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, 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, vice president for Publications, member-at-large on the Board of Governors, chair of the distinguished lecturer Selection Committee, member of IEEE fellow Selection Committee of the ComSoc. He served as the editor-in-chief of the *IEEE IoT Journal*, *IEEE Network*, and *IET Communications*.