future internet

MDPI

Article

Multi-Agent Deep Reinforcement Learning-Based Fine-Grained Traffic Scheduling in Data Center Networks

Huiting Wang [1], Yazhi Liu [1,*], Wei Li [1] and Zhigang Yang [2]

[1] College of Artificial Intelligence, North China University of Science and Technology, Tangshan 063210, China; 18322498530@163.com (H.W.); lw@ncst.edu.cn (W.L.)
[2] College of Electrical Engineering, North China University of Science and Technology, Tangshan 063210, China; yzg@ncst.edu.cn
* Correspondence: liuyazhi@ncst.edu.cn

Abstract: In data center networks, when facing challenges such as traffic volatility, low resource utilization, and the difficulty of a single traffic scheduling strategy to meet demands, it is necessary to introduce intelligent traffic scheduling mechanisms to improve network resource utilization, optimize network performance, and adapt to the traffic scheduling requirements in a dynamic environment. This paper proposes a fine-grained traffic scheduling scheme based on multi-agent deep reinforcement learning (MAFS). This approach utilizes In-Band Network Telemetry to collect real-time network states on the programmable data plane, establishes the mapping relationship between real-time network state information and the forwarding efficiency on the control plane, and designs a multi-agent deep reinforcement learning algorithm to calculate the optimal routing strategy under the current network state. The experimental results demonstrate that compared to other traffic scheduling methods, MAFS can effectively enhance network throughput. It achieves a 1.2× better average throughput and achieves a 1.4–1.7× lower packet loss rate.

Keywords: data center network; traffic scheduling; multi-agent deep reinforcement learning; in-band network telemetry; programmable data plane

Citation: Wang, H.; Liu, Y.; Li, W.; Yang, Z. Multi-Agent Deep Reinforcement Learning-Based Fine-Grained Traffic Scheduling in Data Center Networks. *Future Internet* **2024**, *16*, 119. https://doi.org/10.3390/fi16040119

Academic Editor: Gianluigi Ferrari

Received: 4 March 2024
Revised: 20 March 2024
Accepted: 29 March 2024
Published: 31 March 2024

1. Introduction

With the development of modern networks and the rise of emerging technologies such as cloud computing, data centers carry most of the current data traffic and application functions on the Internet. The diversification of data center applications, the rapid increase in network data volume, and the continuously growing user demands can lead to network congestion and fail at timely forward processing [1]. Therefore, achieving effective management of internal traffic in data center networks (DCNs) and the efficient utilization of network resources are critical challenges that data centers need to address. How to combine traffic characteristics, linking status and application requirements to improve the performance of data center networks, is of great research significance for designing reasonable and efficient traffic scheduling strategies for DCNs.

A Software-Defined Network (SDN) [2] is a novel network model that separates the data plane from the control plane. The data plane is responsible for high-speed forwarding while its control functions are integrated into the controller. With an SDN, a targeted traffic scheduling model can be devised, providing a new approach to solve the traffic scheduling issues of the data center. The use of In-band Network Telemetry (INT) technology can provide fine-grained real-time monitoring of packet paths and the processing process in the network. Based on real-time monitoring information provided by INT, a DCN can achieve precise traffic scheduling and dynamically adjust traffic routing to optimize network performance and resource utilization.

With the in-depth research and application of artificial intelligence technology in the field of computer networks, traffic scheduling research based on deep reinforcement

learning (DRL) [3] has achieved remarkable results in this field. Compared with the traditional routing algorithms, the intelligent routing model that combines a SDN with deep reinforcement learning can automatically learn the mapping relationship between input parameters and output results by observing the empirical information provided by the environment. This not only saves time and costs but also continuously adjusts to output better routing strategies as the environment changes so as to achieve the purpose of load balancing. In recent years, the proposal of multi-agent deep reinforcement learning (MADRL) [4] has brought new hope for optimizing network performance. This method solves large-scale complex problems based on the relationship between individual agents collaborating with each other, which not only reduces the complexity of problem-solving for individual agents but also improves the overall problem-solving ability of multiple agents. In summary, by setting the network topology, the data packet processing logic is realized on the programmable data plane using P4 language, and the intelligent machine learning method is introduced on the control plane to effectively complete the routing decision and the corresponding forwarding of the data packet. Therefore, applying the MADRL method to a data center network will provide a new strategy to solve the load balancing problem.

This paper proposes a traffic scheduling method called MAFS based on multi-agent deep reinforcement learning. In-Band Network Telemetry was used to collect global network state information [5], and the method customizes the environment, state space, action space, and reward function of the model. Furthermore, it designs the CTN-MADDPG algorithm based on a one-dimensional convolutional neural network (1D-CNN) [6] and long short-term memory network (LSTM) [7] to realize traffic scheduling decisions centered on the control plane.

In general, the main contributions of this proposed work include the following:

A traffic scheduling method based on multi-agent deep reinforcement learning is proposed. In the process of traffic scheduling, the CTN-MADDPG algorithm is designed, enabling individual agents to learn how to better collaborate in the network environment to complete routing strategies through the joint training of deep neural networks for agents.

In the CTN-MADDPG algorithm, the 1D-CNN and LSTM are combined to design the Actor network and Critic network for each agent so that the model captures time-dependent relationships in sequential data, enhancing the modeling and representation capabilities of the model for time-series data and optimizing the selection of routing strategies.

The multi-step experience replay strategy is adopted in the CTN-MADDPG algorithm, which is an extension of experience replay. It allows for agents to update the routing strategy by using the experience of multiple consecutive time steps during the training process. This approach maximizes the utilization of temporal correlation and improves the learning efficiency of the model for further optimization of the routing strategy.

2. Related Works

The traffic scheduling of a DCN is used to solve the problem of when and at what rate each data flow in the network should be transmitted. DCNs require efficient, flexible, and secure traffic scheduling methods to optimize routing for different types of traffic. Currently, there are numerous research efforts focused on traffic scheduling in DCNs.

Traditional traffic scheduling methods include the Shortest Path First (SPF) algorithm, the Equal-cost Multi-path (ECMP) [8] algorithm, and the heuristic algorithms. The SPF algorithm is simple, but it has the drawback of being unable to fully utilize network resources. As a traditional static traffic scheduling method, ECMP lacks congestion awareness and may map multiple large flows onto the same path, resulting in congestion in the network. The current dynamic traffic scheduling algorithms, such as FDALB [9] and DIFFERENCE [10], primarily focus on large flows. The dynamic traffic scheduling based on SDNs on links with congestion or load imbalance can effectively reduce the probability of link congestion and improve the transmission performance of the traffic. Among them,

scheduling schemes such as Hedera [11], Mahout [12], Nimble [13], and Fincher are often used to distinguish between large and small streams.

Heuristic algorithms [14] are proposed relative to optimization algorithms, which are generally used to solve non-deterministic polynomial problems, providing approximate optimal solutions. The swarm intelligence algorithm is a heuristic algorithm used to find the global optimal solution. In recent years, it has been commonly used in data center network traffic scheduling technology to find the optimal scheduling route for network traffic efficiently and quickly. For example, Ant Colony Optimization [15] and Particle Swarm Optimization, which belong to swarm intelligence algorithms, have been combined with traffic scheduling problems. However, these algorithms also have certain problems and shortcomings, such as complex computation, a slow convergence speed in the later stage, an inability to simultaneously achieve the calculation speed and result accuracy, a susceptibility to local optima, and an inability to guarantee the global optimality of solutions. They are unable to cope with rapid network traffic changes and therefore cannot perfectly meet the requirements of network performance.

In recent years, with the rapid development of machine learning technology, researchers have introduced and applied it to SDNs based on the effectiveness of machine learning technology in policy decision-making for routing scheduling issues in complex networks and conducted many related studies. Mammeri [16] provided a comprehensive overview of the research on utilizing reinforcement learning (RL) methods to solve routing problems, summarizing commonly used reinforcement learning models and the current challenges. Li Ying et al. [17–20] proposed a Q-Learning based routing algorithm for sensor networks and opportunistic networks. When learning convergence is difficult due to the increase in the state-space dimension, using neural networks to estimate the Q-value can solve the problem of dimension explosion while reducing the average latency and improving network throughput.

Reference [21] introduced RL neural networks into the QoS routing calculation; it proposed a variable greedy function and improved the reward function to optimize the routing decision model. This routing scheme consumes excessive controller resources and has long computation times, resulting in poor timeliness. Zuo Y et al. [22] transformed the path planning problem into a sequence inference problem of network nodes, using a sequence-to-sequence model to learn forwarding paths based on network traffic. These approaches are centered on a single node and do not consider traffic scheduling problems from the perspective of the overall network.

Reference [23] utilized DRL agents with convolutional neural networks to improve the performance of the routing configuration in complex networks and enhance the performance of QoS-aware routing. Reference [24] proposed an off-line training routing solution combining the Deep Q-Network (DQN) and LSTM algorithms, which used batch reinforcement learning methods to learn the optimal control strategies from pre-collected transition samples without interacting with the system. Reference [25] employed a framework combining imitation learning and deep reinforcement learning, effectively reducing the instability of RL algorithms.

After several years of development, multi-agent deep reinforcement learning technology is one of the important technological approaches to solve decision control problems in complex environments. It has been successfully applied in various fields, such as game confrontation, robot obstacle avoidance, drone formation control, and traffic signal management. Gradually, it is becoming a key method to study the emergence of swarm intelligence in multi-agent systems.

Kim et al. [26] introduced the Medium-Access Control (MAC) methods from the communication domain into multi-agent deep reinforcement learning, proposed a Scheduled Communication model, which optimized the transmission patterns of information and enabled agents to have full-time communication capability. A multi-agent team from Tsinghua University combined the Minimax principle with the MADDPG algorithm to propose the M3DDPG algorithm [27]. The Minimax principle is used to estimate the worst-case

scenario where the behaviors of all agents in the environment are completely hostile, and the agent strategies are continuously updated based on this estimated worst-case scenario, which improves the robustness of the agent learning strategies and ensures the effectiveness of learning. Reference [28] proposed a real-time distributed learning method in mobile edge networks. It employed an MADRL model to realize the independent routing decisions for each edge node, overcoming the limitations of traditional turn-based DRL methods and improving the transmission packet ratio and effective throughput.

In summary, the feature expression capabilities and autonomous learning mechanisms of the multi-agent deep reinforcement learning model can be used to calculate the optimal routing strategies for traffic and perform real-time forwarding. Therefore, this paper proposes the traffic scheduling method MAFS, which intelligently realizes the packet processing logic of the data plane and the centralized decision-making of the control plane.

3. System Scheme and Implementation

3.1. MAFS Traffic Scheduling System

This paper proposes the CTN-MADDPG algorithm based on the MADDPG algorithm and builds the MAFS traffic scheduling system. The overall structure is shown in Figure 1. The core of this system model is the control plane, where the controller obtains information from the underlying network through a southbound interface to achieve centralized control function. The current network state information is obtained using INT on the data plane and transmitted to the control plane. The control plane preprocesses the network state data, using it as input for a neural network to calculate the routing strategies. Subsequently, the controller converts the optimal routing strategy into flow table rules, which are then deployed to the data plane.

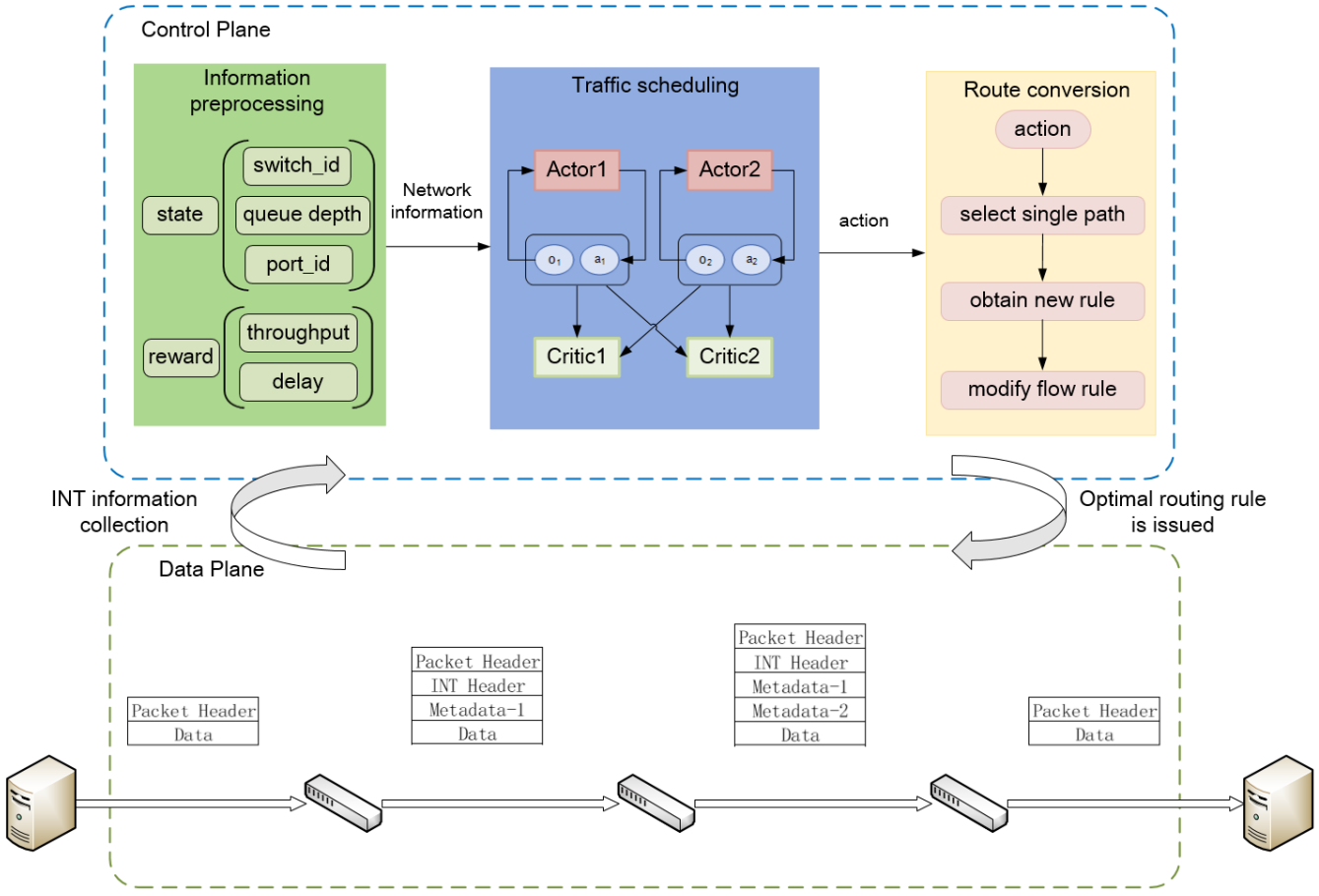

Figure 1. Architecture of MAFS traffic scheduling scheme.

Information Preprocessing Module: This module is primarily responsible for processing the obtained information from the data plane and feeding the processed data into the traffic scheduling module. It integrates the switch load information collected by INT within the data plane as input to the state information of the Actor network and Critic network. Additionally, it consolidates the link load information received from the receiver feedback into reward values, which are utilized to guide the selection of the routing strategies for each intelligent agent.

Traffic Scheduling Module: This module primarily executes the CTN-MADDPG algorithm, with each intelligent agent corresponding to an Actor–Critic framework. Each intelligent agent interacts with the data plane environment to obtain the current state information of the agent, and the distributed execution of the Actor network obtains the optimal routing strategy. In the centralized training process, the Critic network can obtain the global state information and the routing strategies adopted by all the intelligent agents. It grasps the global value function to evaluate the utility of the set of routing strategies selected by each agent to guide the update of the Actor network.

Route Conversion Module: This module is primarily used to convert the routing selection results of the traffic scheduling module into the form of flow rules in order to obtain new flow table rules. It dynamically modifies the flow rules executed by the switches on the control plane and distributes them to the data plane. When a data packet enters a switch, the switch will forward it based on the corresponding forwarding rules to achieve dynamic data transmission.

Compared with the previous single-agent deep reinforcement learning methods, the multi-agent deep reinforcement learning method can assign decision tasks to multiple agents for parallel learning and decision-making. Each agent can independently observe network states and make routing decisions. By sharing experiences and learning for each other, agents can improve their routing strategies, thereby improving overall network performance.

The MAFS traffic scheduling method proposed in this paper designs the CTN-MADDPG algorithm in the traffic scheduling module, which is based on the MADDPG algorithm of MADRL. The MADDPG algorithm is a deep deterministic policy gradient algorithm for a multi-agent environment. Each agent corresponds to an Actor–Critic network structure, adopting the mode of centralized training and distributed execution. It combines the advantages of the action-value function and policy gradient method, utilizing the information of the action-value function to guide the policy update, thereby addressing the deterministic policy optimization problem in a continuous action space. In general, the structure of the Actor network and Critic network of the MADDPG algorithm are composed of three fully connected layers, which may not be able to capture complex feature information. In addition, the model has fewer parameters, which can easily lead to overfitting or lower learning efficiency.

The CTN-MADDPG algorithm is based on the MADDPG algorithm model, which combines the one-dimensional convolutional neural network (1D-CNN) and long short-term memory network (LSTM) to form a compact network model and introduces temporal information to design a multi-step experience replay strategy. It can extract the structural and temporal features of network state information. The network state information that occurred in the past time is transferred to the calculation of the present time. Therefore, the action information of the last time can be used to make an effective selection of the next routing action. The CTN-MADDPG algorithm can match the traffic to the optimal path based on real-time network state information, thereby achieving balanced distribution and effective transmission of traffic in the DCN.

3.2. CTN-MADDPG Intelligent Network Model

This paper constructs a hybrid deep reinforcement learning model based on the 1D-CNN and LSTM concatenated for each agent. The Actor–Critic network model structure consists of three network modules: the one-dimensional convolution module, LSTM mod-

ule, and fully connected layer module. CTN-MADDPG utilizes the one-dimensional convolutional neural network layer to learn the features of the network state information and extracts the temporal information of the learned features through the LSTM layer, which can effectively improve the generalization ability and traffic prediction capability of the model and achieve optimal routing and forwarding. The structure of the CTN-MADDPG intelligent network model is shown in Figure 2.

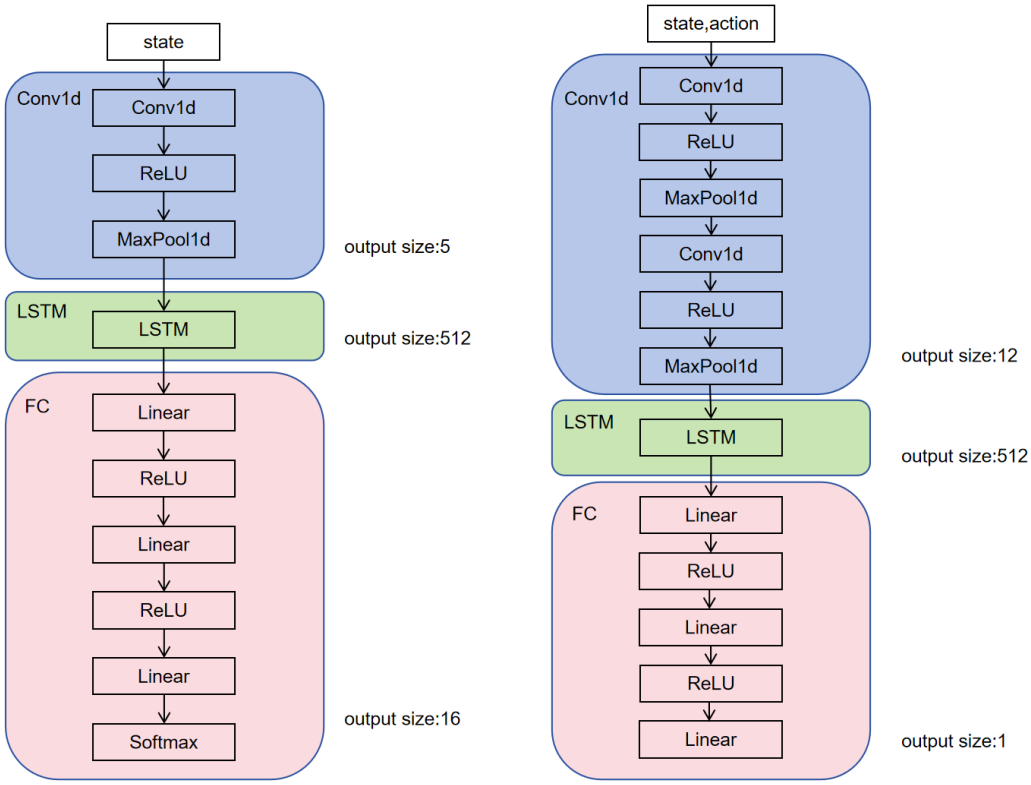

Figure 2. The structure of CTN-MADDPG intelligent network model.

In this algorithm, the one-dimensional convolutional network module is added to the Actor–Critic architecture, which consists of a one-dimensional convolutional layer (Conv1d), max-pooling layer (MaxPool1d), and ReLU activation function layer.

Furthermore, the second network module in the Actor–Critic architecture of this paper is the LSTM module. The LSTM module is used to learn the feature vectors processed by the one-dimensional convolutional layer, exploring their internal rules and obtaining their hidden layer states. By combining LSTM, the effective temporal information from action values and observed states can be introduced and processed. Based on the action information from the previous time step, the actions to be taken at the current time step can be effectively predicted and properly adjusted so as to accelerate the training efficiency of the network and achieve a better control effect.

In this paper, LSTM is applied to the traffic scheduling task of the data center, which makes LSTM learn how to predict the state and reward. Through this approach, it infers that the network states obtained by each agent interact with the network topology environment. At the point, the agent wants to obtain the action- and state-value functions of the model output when forwarding data packets, which depends not only on the current state s but also on the additional auxiliary signal h_{t-1} from the self-propagation of the LSTM memory unit, which contains the key features extracted from the network topology in the previous states. The forward propagation of all states relies on the same LSTM, which runs through the entire episode. The cell state and hidden state transmitted inside the LSTM serve as the long-term memory and short-term memory of the agent, respectively, helping the agent

to better grasp the global information and cope with the problem of hidden states in the current state in real situations.

In this paper, the third network module of the Actor–Critic architecture is the fully connected layer module. The last layer of the Actor network outputs the routing actions. The last layer of the Critic network changes the output dimension to 1, which serves as feedback from the Critic network to the Actor network, guiding the Actor network to update and select the optimal action strategy.

3.3. Multi-Step Experience Replay Strategy

The experience replay buffer stores previous experiences in a buffer so that agents can more efficiently utilize these experiences for learning when training neural networks. It can provide training stability in multi-agent scenarios.

Based on experience replay, agents not only update the parameters of neural networks using the current interaction experiences but also conduct batch learning by randomly sampling a batch of past experiences from the buffer. This helps to avoid the model only learning the current specific contexts and improve sample efficiency, that is, using a limited number of interactions and sampling through reasonable strategies to interact with the environment, and a more comprehensive exploration of the environment is achieved to optimize strategies.

The experience information randomly sampled during the experience replay can be represented by a tuple (s_t, a_t, r_t, s_{t+1}), including the current state, action, reward, and next state. However, randomly sampling only one set of network information may disrupt the temporal correlation in the state sequence. In traffic scheduling tasks, the network state information at certain moments may be closely related to previous or subsequent network states. If only the information from the current state is randomly sampled, it may lose this temporal structure, thereby affecting the learning effect. In some cases, a continuous sequence of states may contain more information about the environment, and random sampling taking only one set of state information may not capture this continuity.

Therefore, this paper sets up a multi-step experience replay strategy, which considers sampling a continuous sequence of network states rather than just single network state points. Retaining a certain number of continuous state experiences allows for making more use of temporal information and capturing more contextual information. The structure of the multi-step experience replay buffer is shown in Figure 3.

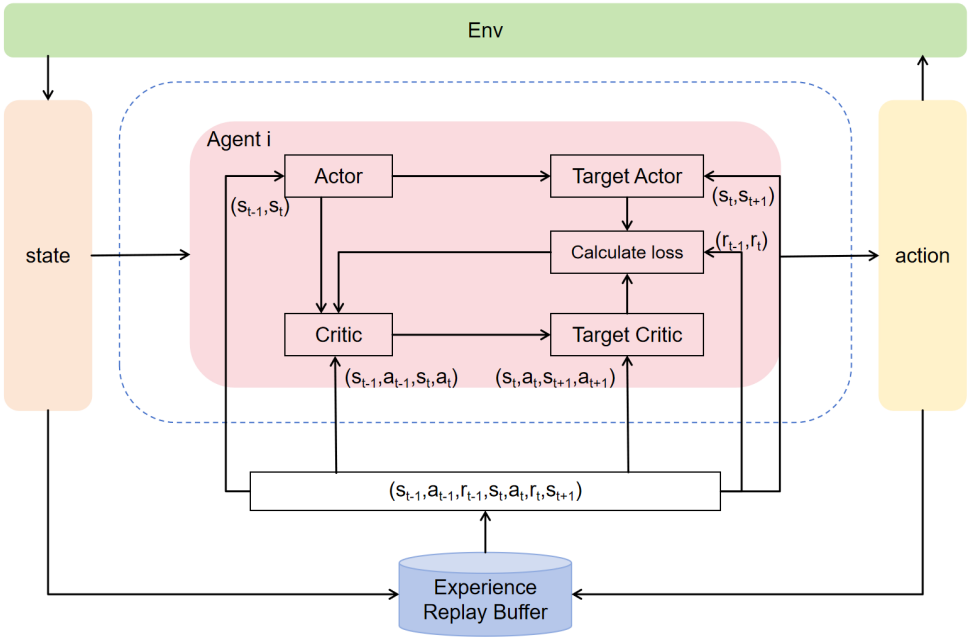

Figure 3. Multi-step experience replay of MAFS.

The MAFS method introduces temporal correlation information by extending the experience tuples stored in the experience replay buffer. Add the previous time-step state, action, and reward of the agent to the experience information, forming the new tuple as follows:

$$(s_{t-1}, a_{t-1}, r_{t-1}, s_t, a_t, r_t, s_{t+1}), \tag{1}$$

The optimized experience information is stored in the experience replay buffer and sampled randomly. In order to improve the computational performance, effective dimensionality reduction and consolidation of the data are conducted. In the distributed execution of the CTN-MADDPG algorithm, the input data of the Actor network is the network state information from the previous time step and the current time step. The Critic network not only inputs the current state information and routing strategy of all agents but also includes information from the previous state. In the centralized training process, the input information of the Target Actor network includes the network state information of the current moment and next moment so as to output the routing strategy of the next time step, which is then passed to the Target Critic network to update the network parameters. The input information in the Target Critic network includes the current and next state information of all the agents, as well as the routing strategies they have made. Temporal features are introduced in the training process of the CTN-MADDPG algorithm to optimize the updating process of the network parameters. By extending and optimizing the experience information stored and sampled from the experience replay buffer, multi-agent systems can more effectively utilize past experiences, thereby accelerating the convergence process of deep reinforcement learning.

Considering the temporal relationships between states, the simultaneous use of random sampling and temporal experience cannot only ensure the temporality of the neural network model training under the current state but can also be used in conjunction to enhance the stability of the deep reinforcement learning model training, thereby improving the sample efficiency.

4. Implementation of MAFS Traffic Scheduling Algorithm

4.1. Key Parameter Setting

The CTN-MADDPG algorithm is deployed on the control plane, and the network state information is first collected by INT and defined as the input data for the network model. The network topology environment is constructed based on the traffic scheduling scenarios in the DCN. The key elements of the CTN-MADDPG algorithm are designed, namely, the state space, action space, and reward function. The trained CTN-MADDPG algorithm can determine the optimal routing path through the current network environment.

(1) State space: The network state obtained between two terminal hosts through INT. The network state is defined as the load information of the switches in the current link.

$$s = (o_1, o_2, \cdots, o_n), \tag{2}$$

$$o_i = \{swid_i, qdepth_i, port\_id_i\}, \tag{3}$$

where o_i represents the ith observed state, and each observed state is composed of a set of switch identifiers, queue depths, and output port numbers for each switch. It is the global state information obtained after integrating the state information of individual agents.

(2) Action space: The set of shortest paths that can be scheduled between the source node and the destination node. When selecting an action a_i, the ith path among the n shortest paths is preferred.

$$a = (a_1, a_2, \cdots, a_n), \tag{4}$$

(3) Reward: The reward value obtained through training the neural network based on the current environment and actions taken. The reward function for each agent is set as follows:

$$r_i = \log t_i - \log d_i, \tag{5}$$

where t_i represents the average throughput in the ith observed state, which focuses on the data transmission capacity of the network. d_i represents the delay in the ith observed state, which focuses on the time that the current packet experiences in the links. Considering both these two indicators can provide a comprehensive evaluation of the network performance, and the routing strategies based on these indicators can help to optimize the network and improve the overall performance. By using the logarithmic function to converge parameter values to a certain range, the magnitude difference between the two parameters can be narrowed to balance the influence of different parameters, which helps to predict the network performance more accurately.

4.2. The Algorithm of MAFS

During the operation of MAFS, the SDN controller monitors the global network states in real time by establishing a connection with the switches for data interaction. Through continuous iterative training and updating, the CTN-MADDPG algorithm can optimize its own network model and continuously learn network states to generate the optimal forwarding paths, thereby achieving load balancing.

The CTN-MADDPG algorithm assumes that there are N agents, where the policy representation of each agent is π_i, which is the mapping of the network state information to the routing policy. The policy set is $\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$, and the set of policy parameters is $\theta = \{\theta_1, \theta_2, \cdots, \theta_N\}$. The computation process of MAFS is outlined in Algorithm 1.

Algorithm 1 MAFS Method

Require: Data plane information collected by INT;
Ensure: Optimal path;
1: Set the state and action; initialize Actor Network parameters $\theta = \{\theta_1, \theta_2, \cdots, \theta_N\}$ and Critic Network parameters $\phi = \{\phi_1, \phi_2, \cdots, \phi_i\}$ for each agent;
2: **for** *episode* $= 1$ to $MAX\_EPISODE$ **do**
3: Reset the environment, obtain initial state s_t;
4: **for** *step* $t = 1$ to MAX_STEP **do**
5: **for** *agent* $i = 1$ to N **do**
6: action $\leftarrow$ mafs.action(s);
7: Select scheduling path as a_i, and controller execute the a_i;
8: **end for**
9: Observe reward r_t and next state s_{t+1};
10: $s_t \leftarrow s_{t+1}$;
11: Store $(s_{t-1}, a_{t-1}, r_{t-1}, s_t, a_t, r_t, s_{t+1})$ in experience replay buffer D;
12: **for** *agent* $i = 1$ to N **do**
13: Sample a random minibatch of S samples from D;
14: Set $y = r_i + \gamma Q_i^{\mu'}(s, s', a_1, \cdots, a_N, a'_1, \cdots, a'_N)|_{a'_j = \mu'_j(o_j)}$;
15: Update Critic Network by minimizing the loss $L(\theta_i)$;
16: Update Actor Network using the sampled policy gradient $\nabla_{\theta_i} J(\mu_i)$;
17: **end for**
18: Update target network parameters for each agent i: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$; $\phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i$:
19: **end for**
20: **end for**

The MAFS method updates the network model by providing a reward indicator of the current data center network load balancing based on the current network state s_t and the selected scheduling path a_t. By maximizing the cumulative reward, the load in the data center network becomes more balanced. The cumulative expected reward for each agent is given by:

$$J(\theta_i) = \mathbb{E}_{s\sim\rho^{\pi}, a_i\sim\pi_{\theta_i}}\left[\sum_{t=0}^{\infty}\gamma^t r_i, t\right], \tag{6}$$

where θ_i represents the parameters in the ith policy network, and r_i represents the reward value of the ith agent, which is composed of the average throughput and end-to-end delay in the current network.

In the process of solving the traffic scheduling problem in the DCN, if the random policy is adopted, and the selection of the action is made according to the probability distribution, the process of policy optimization will be affected by large fluctuations and lead to the instability of the training. In contrast, the deterministic policy based on the CTN-MADDPG algorithm enables each agent to choose a specific action based on the current network environment information to make a specific routing decision, which makes the agent have better stability and convergence in the process of making a traffic scheduling decision. At the same time, the deterministic policy can more flexibly adjust the traffic scheduling strategies to adapt to different network topology environments and traffic load situations, thus improving the global network performance.

Therefore, the deterministic policy is adopted in this paper. In a given network state, each agent directly outputs a complete routing path based on the current network environment information. This makes the behavior of agents more stable and controllable, helping to coordinate the behavior of each agent and enabling better routing strategy optimization and path selection. In the case of deterministic strategies μ_{θ_i}, the policy gradient is given by:

$$\nabla_{\theta_i} J(\mu_i) \approx \frac{1}{S}\sum_{j}\nabla_{\theta_i}\mu_i(a_i|o_i)\nabla_{a_i}Q_i^{\mu}(s, a_1, \cdots, a_N)|_{a_i=\mu_i(o_i)}, \tag{7}$$

where o_i represents the observation information of the ith agent, and $s = [o_1, o_2, \cdots, o_N]$ denotes the observation vector, which is the network state information. The routing policy of each agent is updated through gradient descent, and the resulting routing policy is improved by interacting with other agents.

The above gradient formula adopts the experience replay technique to extend the experience replay buffer, which contains several pieces of information, with its elements represented as a tuple $(s_{t-1}, a_{t-1}, r_{t-1}, s_t, a_t, r_t, s_{t+1})$.

The Actor network is updated using the above gradient descent method, and the traffic in the data center network is forwarded through the routing policy selected by the Actor network. The Critic network is updated by the loss function through backpropagation. Its input is the global information about the network state and routing actions, with the loss function calculation formula as follows:

$$L(\theta_i) = \frac{1}{S}\sum\left(Q_i^{\mu}(s, s', a_1, \cdots, a_N, a_1', \cdots, a_N') - y\right)^2, \tag{8}$$

where $y = r_i + \gamma Q_i^{\mu'}(s, s', a_1, \cdots, a_N, a_1', \cdots, a_N')|_{a_j'=\mu_j'(o_j)}$, $Q_i^{\mu'}$ represents the target network, and $\mu' = \{\mu_1', \cdots, \mu_N'\}$ is the set of target policies. The policies of other agents can be obtained through fitting approximation without the need for further communication interaction.

The process of multi-agent systems training is similar to that of single-agent training. A centralized Critic network can alleviate the difficulty of convergence caused by the dynamic environmental changes in multi-agent scenarios. After the training is completed, each Actor network can adopt the corresponding routing actions based on its own network-observed states.

5. Experiment

5.1. Experimental Environment Setting

Based on the MADDPG algorithm, this paper designs an intelligent network model and a multi-step experience replay mechanism on the basis of the original code [29]. The CTN-MADDPG algorithm is proposed, and the MAFS traffic scheduling method is implemented by designing functional modules on the control plane and data plane.

The description of the simulation experiment environment of the traffic scheduling method in the programmable data plane is shown in Table 1.

Table 1. Descriptions of simulation experiment environment.

Simulation Tool	Description
Operating System	Ubuntu 18.04 64-bit
Network Simulation Platform	Mininet 2.2.2
Virtual Switch	Bmv2 Software Switch
Programming Languages	Python, P4
Traffic Generation Tool	iperf
Python Management Software	Anaconda

The specific parameter values of the network model structure in the MAFS traffic scheduling method are shown in the following Table 2.

Table 2. Network model parameter values of MAFS.

Parameter Name	Value	Description
BATCH _SIZE	1024	Size of mini-batch sampling
MEMORY _CAPACITY	10,000	Capacity of buffer pool
MAX _EPISODE	500	Maximum number of episodes
MAX _STEP	25	Maximum number of steps per episode
LR _A	0.01	Actor network learning rate
LR _C	0.02	Critic network learning rate
GAMMA	0.99	Discount factor
TAU	0.01	Target network hyperparameter

5.2. Experimental Results and Analysis

In order to verify the performance of MAFS, this paper compares it with the ECMP algorithm and traffic scheduling schemes based on DDPG and MADDPG through an experimental simulation and analyzes the experimental results under different performance indexes. To analyze the differences between the different schemes, the link bandwidths are adjusted to 100 Mbps, 500 Mbps, 1000 Mbps, and 2000 Mbps, respectively.

The experimental results indicate that under different bandwidth settings, the average throughput of MAFS is higher than that of the ECMP, DDPG, and MADDPG traffic scheduling schemes, while the average round-trip delay, packet loss rate, and flow completion time of MAFS are lower than the other three comparison methods.

The average throughput is used to evaluate network performance, which represents the capability of the traffic in the network as it is forwarded along paths. As depicted in Figure 4, the comparison of the average throughput shows that compared to the ECMP, DDPG, and MADDPG methods, the MAFS method increases the average throughput by 54.19%, 24.79%, and 16.93% when the bandwidth is set to 100 Mbps, respectively. These results indicate that the proposed MAFS method can update routing strategies through

continuous learning historical experience and training, selecting the optimal forwarding path for the traffic, thereby achieving network load balancing.

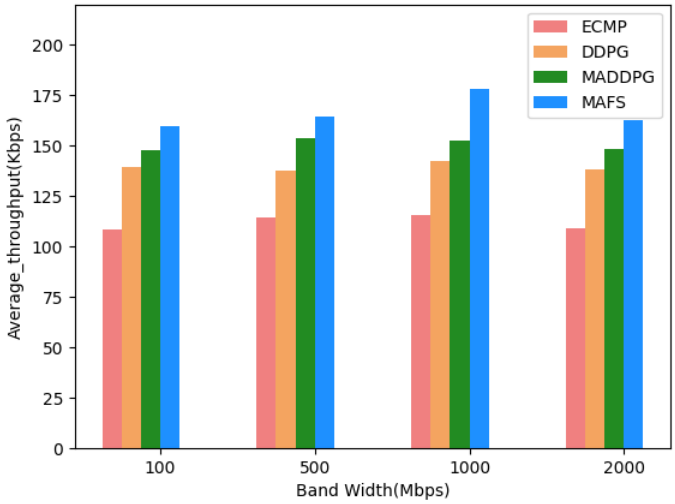

Figure 4. Average network throughput.

The comparison of the average round-trip delay is shown in Figure 5. When the bandwidth is set to 1000Mbps, the average round-trip delay of the MAFS method is reduced by 33.30%, 24.27%, and 19.70%, respectively, compared with the ECMP, DDPG, and MADDPG methods. These results demonstrate that the MAFS method can effectively reduce the average round-trip delay, improve the traffic transmission efficiency, and alleviate link congestion.

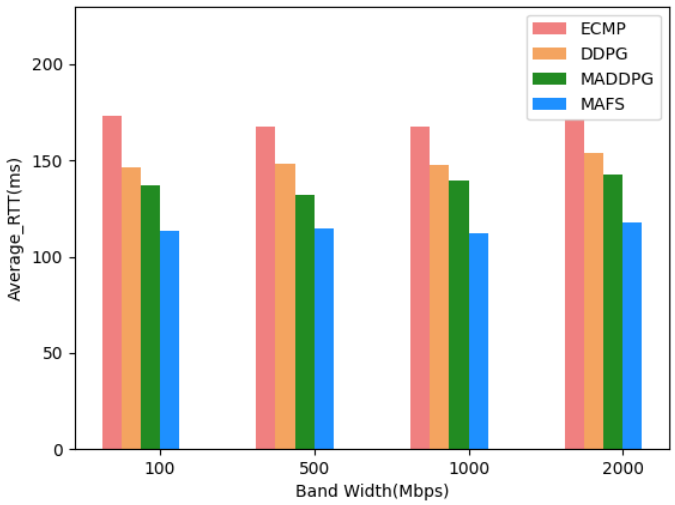

Figure 5. Average round-trip delay.

The comparison of the packet loss rate is shown in Figure 6. Under different bandwidth settings, the MAFS method can effectively control the packet loss rate within a small range. Compared with ECMP, DDPG, and MADDPG, the loss rate of the MAFS method is kept below 15.54%, which can intelligently allocate forwarding paths for traffic, alleviate link loads to a certain extent, effectively reduce the packet loss rate, and ensure the effective transmission of packets.

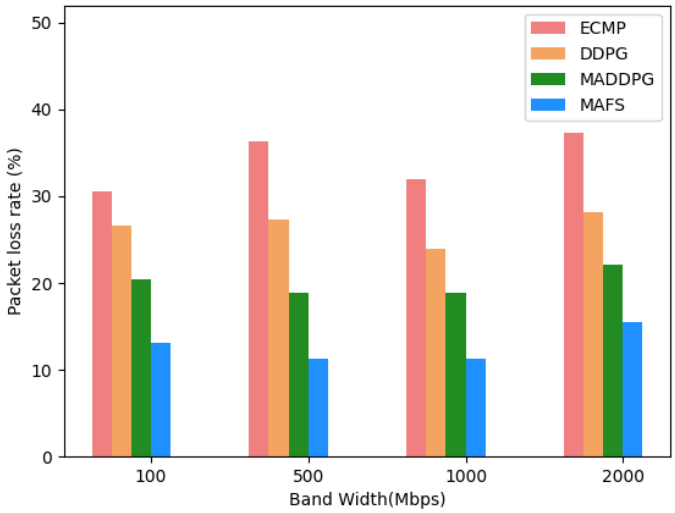

Figure 6. Packet loss rate.

As shown in Figure 7, with different bandwidth settings, the MAFS method can effectively reduce the flow completion time. Particularly, when the bandwidth is set to 100Mbps, the flow completion time of the MAFS method is reduced by 36.84%, 26.47%, and 13.46% compared with the ECMP, DDPG, and MADDPG methods, respectively. The results demonstrate that the MAFS method can achieve faster data transmission, thereby making more efficient use of the available bandwidth to improve network utilization.

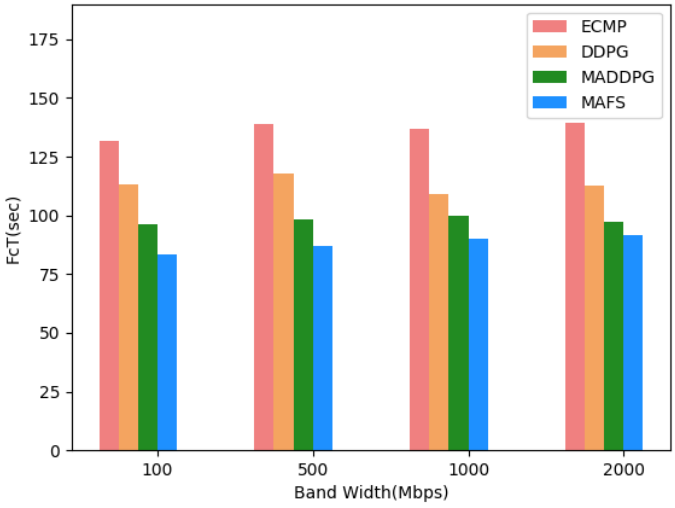

Figure 7. Flow completion time.

Figure 8 shows the execution time of the intelligent algorithm for the MAFS traffic scheduling scheme and other comparative traffic scheduling methods during the network model training process. Specifically, it starts from the agent obtaining network state information and continues until a new routing strategy is calculated through an intelligent network model and converted into flow rules for issuance. The experimental results show that the CTN-MADDPG traffic scheduling algorithm of the MAFS method has a shorter execution time, which means that it can complete the model learning and routing decision faster and can respond more quickly to the changes in the network environment to reduce the routing decision delay. Therefore, the MAFS traffic scheduling method has higher computational efficiency.

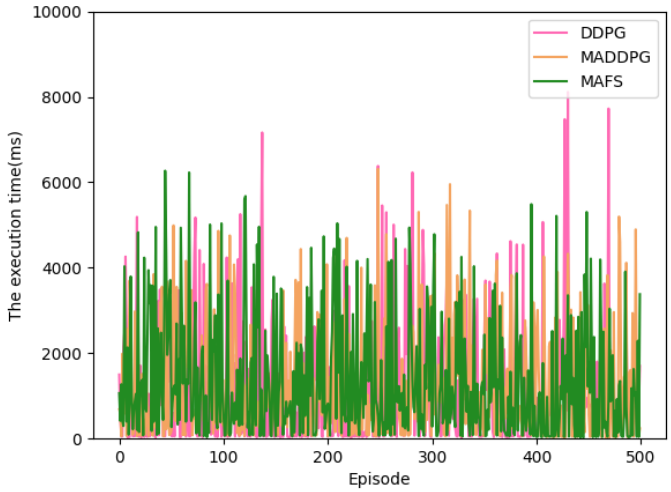

Figure 8. The execution time varies with the episodes.

Figure 9 records the changes in the throughput of MAFS, DDPG, and MADDPG with the training episodes. In the early stages of system operation, the throughput changes in DDPG, MADDPG, and MAFS are not very different due to the limited number of algorithm training steps. With the increase in the training period, the network throughput of MAFS is significantly better than the other two traffic scheduling methods, and the throughput continues to improve and tends to stabilize. Figure 10 records how the delay varies for MAFS, DDPG, and MADDPG with the training episodes. Compared with the DDPG and MADDPG algorithms, the delay of MAFS is kept at a small value, which steadily decreases and stabilizes over time. Figure 11 records the changes in the reward values of MAFS, DDPG, and MADDPG with the training episodes. Although the reward values of the MAFS method in the training process have a certain jitter, they are relatively stable after 425 episodes and reach a convergence state. Various agents cooperate with each other to complete the task of maximizing the network utility.

The experimental results demonstrate that the MAFS traffic scheduling method can effectively reduce network delay and improve network throughput with good convergence and stability.

Based on the above results analysis, the MAFS method can formulate optimal traffic scheduling strategies based on real-time state information. Compared with the ECMP, DDPG, and MADDPG methods, it can effectively improve network performance and avoid congestion in the network to ensure effective data transmission and achieve load balancing.

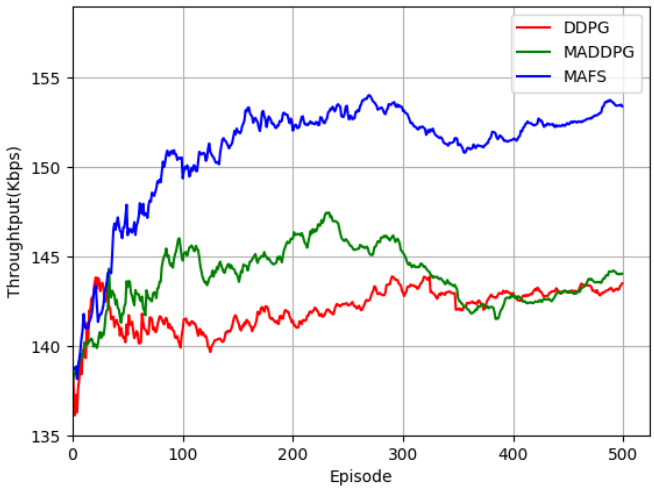

Figure 9. Throughput varies with the episodes.

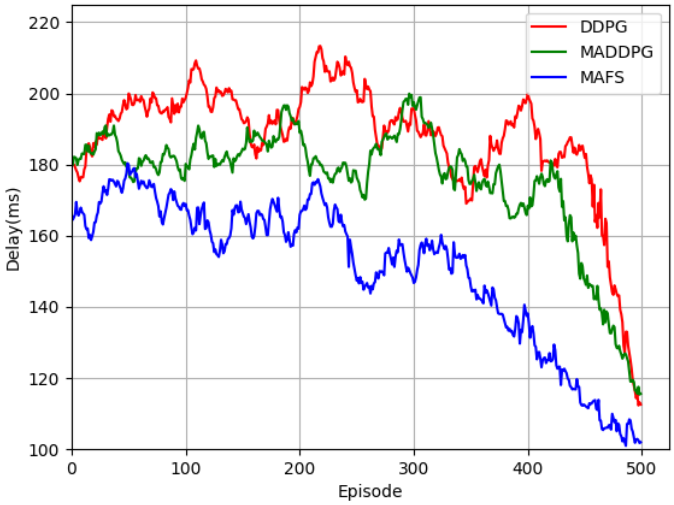

Figure 10. Delay varies with episodes.

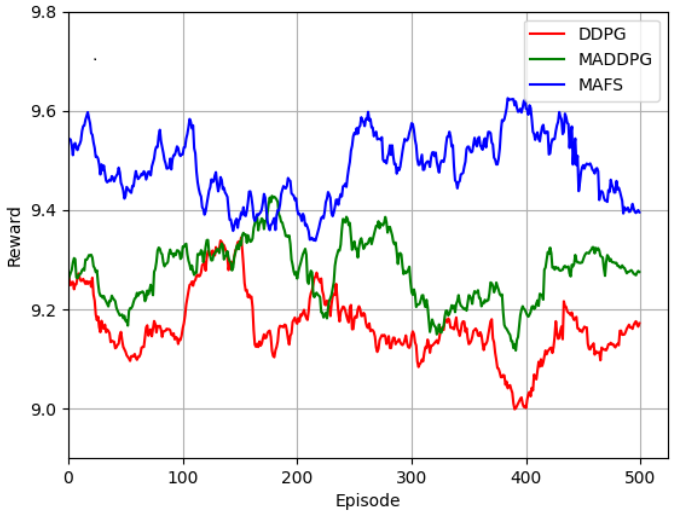

Figure 11. Reward varies with episodes.

6. Conclusions

Comprehensively considering the influence of the queue depth, throughput, and delay factors on routing decisions, this paper proposes a traffic scheduling scheme based on multi-agent deep reinforcement learning. In this paper, the traffic scheduling problem is modeled. The system architecture of the MAFS intelligent traffic scheduling scheme is designed, and the network model is designed to optimize the network model by combining the one-dimensional convolutional neural network and long short-term memory neural network. In addition, the multi-step experience replay mechanism is used to make traffic scheduling strategies based on continuous network state information. The performance testing results of the proposed algorithm on the experimental platform show that compared with the existing traffic scheduling schemes, the proposed method can effectively alleviate network congestion and achieve higher throughput with a lower delay and packet loss rate.

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

Funding: This work was funded by the Science and Technology Project of Hebei Education Department ZD2022102.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors gratefully appreciate the anonymous Reviewers for their valuable comments.

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

References

1. Chen, Q.; Giambene, G.; Yang, L.; Fan, C.; Chen, X. Analysis of Inter-Satellite Link Paths for LEO Mega-Constellation Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2743–2755. [CrossRef]
2. Bhardwaj, S.; Panda, S.N. Performance evaluation using RYU SDN controller in software-defined networking environment. *Wirel. Pers. Commun.* **2022**, *122*, 701–723. [CrossRef]
3. PEI, P. Integrated Guidance and Control for Missile Using Deep Reinforcement Learning. *J. Astronaut.* **2021**, *42*, 1293.
4. Sun, Y.; Cao, L.; Chen, X. Overview of multi-agent deep reinforcement learning. *Comput. Eng. Appl.* **2020**, *56*, 13–24.
5. Kim, C.; Sivaraman, A.; Katta, N.; Bas, A.; Dixit, A.; Wobker, L.J. In-band network telemetry via programmable dataplanes. In Proceedings of the ACM SIGCOMM, London, UK, 17–21 August 2015; pp. 1–2.
6. Mao, B.; Tang, F.; Fadlullah, Z.M.; Kato, N. An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems. *IEEE Trans. Emerg. Top. Comput.* **2019**, *9*, 1554–1565. [CrossRef]
7. Novaes, M.P.; Carvalho, L.F.; Lloret, J.; Proenca, M.L. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access* **2020**, *8*, 83765–83781. [CrossRef]
8. Ye, J.-L.; Chen, C.; Chu, Y.H. A weighted ECMP load balancing scheme for data centers using P4 switches. In Proceedings of the 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, Japan, 22–24 October 2018; pp. 1–4.
9. Wang, S.; Zhang, J.; Huang, T.; Pan, T.; Liu, J.; Liu, Y. Flow distribution-aware load balancing for the datacenter. *Comput. Commun.* **2017**, *106*, 136–146. [CrossRef]
10. Zhang, H.; Tang, F.; Barolli, L. Efficient flow detection and scheduling for SDN-based big data centers. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1915–1926. [CrossRef]
11. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: dynamic flow scheduling for data center networks. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 28–30 April 2010; pp. 89–92.
12. Curtis, A.R.; Kim, W.; Yalagandula, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proceedings of the 2011 Proceedings IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 1629–1637.
13. Zhang, Y.; Cui, L.; Zhang, Y. A stable matching based elephant flow scheduling algorithm in data center networks. *Comput. Netw.* **2017**, *120*, 186–197. [CrossRef]
14. Li, C.; Jiang, K.; Luo, Y. Dynamic placement of multiple controllers based on SDN and allocation of computational resources based on heuristic ant colony algorithm. *Knowl.-Based Syst.* **2022**, *241*, 108330. [CrossRef]
15. Park, J.; Hwang, J.; Yeom, K. Nsaf: An approach for ensuring application-aware routing based on network qos of applications in sdn. *Mob. Inf. Syst.* **2019**, *2019*, 3971598. [CrossRef]
16. Mammeri, Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access* **2019**, *7*, 55916–55950. [CrossRef]
17. Ying, L.I.; Fang, W.; Dong-Sheng, J.; Fei, Z. A Q-learning Based Routing Approach for Wireless Sensor Network. *Comput. Technol. Autom.* **2017**. [CrossRef]
18. Zhou, Y.; Cao, T.; Xiang, W. Anypath routing protocol design via Q-learning for underwater sensor networks. *IEEE Internet Things J.* **2020**, *8*, 8173–8190. [CrossRef]
19. Zhang, Y.; Zhang, Z.; Chen, L.; Wang, X. Reinforcement learning-based opportunistic routing protocol for underwater acoustic sensor networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2756–2770. [CrossRef]
20. Dhurandher, S.K.; Singh, J.; Obaidat, M.S.; Woungang, I.; Srivastava, S.; Rodrigues, J.J. Reinforcement learning-based routing protocol for opportunistic networks. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
21. Yuan, Z.; Zhou, P.; Wang, S.; Zhang, X. Research on routing optimization of SDN network using reinforcement learning method. In Proceedings of the 2019 2nd International Conference on Safety Produce Informatization (IICSPI), Chongqing, China, 28–30 November 2019; pp. 442–445.
22. Zuo, Y.; Wu, Y.; Min, G.; Cui, L. Learning-based network path planning for traffic engineering. *Future Gener. Comput. Syst.* **2019**, *92*, 59–67. [CrossRef]
23. Pham, T.A.Q.; Hadjadj-Aoul, Y.; Outtagarts, A. Deep reinforcement learning based QoS-aware routing in knowledge-defined networking. In *Proceedings of the Quality, Reliability, Security and Robustness in Heterogeneous Systems: 14th EAI International Conference, Qshine 2018, Ho Chi Minh City, Vietnam, 3–4 December 2018*; Proceedings 14; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 14–26.

24. Xu, Z.; Wu, K.; Zhang, W.; Tang, J.; Wang, Y.; Xue, G. PnP-DRL: A plug-and-play deep reinforcement learning approach for experience-driven networking. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2476–2486. [CrossRef]
25. Fang, Q.; Xu, X.; Wang, X.; Zeng, Y. Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning. *CAAI Trans. Intell. Technol.* **2022**, *7*, 167–176. [CrossRef]
26. Kim, D.; Moon, S.; Hostallero, D.; Kang, W.J.; Lee, T.; Son, K.; Yi, Y. Learning to schedule communication in multi-agent reinforcement learning. *arXiv* **2019**, arXiv:1902.01554.
27. Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; Russell, S. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 4213–4220.
28. He, B.; Wang, J.; Qi, Q.; Sun, H.; Liao, J. RTHop: Real-time hop-by-hop mobile network routing by decentralized learning with semantic attention. *IEEE Trans. Mob. Comput.* **2021**, *22*, 1731–1747. [CrossRef]
29. Available online: https://github.com/philtabor/Multi-Agent-Deep-Deterministic-Policy-Gradients (accessed on 12 January 2024).