Deep Reinforcement Learning-Based Intelligent Traffic Scheduling in Software-Defined Networks

Baoxing Xie
The Department of Traffic Information Engineering, Henan College of Transportation, Zhengzhou 450000, China
E-mail: baoxing_xie@hotmail.com

Keywords: deep reinforcement learning; SDN, traffic scheduling; network optimization; algorithm performance

Received: January 20, 2025

With the continuous increase of Internet traffic, traditional network traffic scheduling methods are facing the problems of insufficient efficiency and adaptability. Software - defined networking (SDN) provides flexible control capabilities for network traffic management, and intelligent traffic scheduling algorithms, especially scheduling methods based on deep reinforcement learning (DQN), can dynamically adapt to traffic changes in different network environments. This paper proposes an intelligent traffic scheduling algorithm based on DQN. The DQN - based algorithm effectively manages and optimizes network traffic by continuously interacting with the network environment, making real - time decisions on traffic path selection and resource allocation. It conducts experimental verification in different network scenarios. By comparing with traditional static routing and load balancing algorithms, the experimental results show that the traffic scheduling algorithm based on DQN has obvious advantages in throughput, delay, packet loss rate and load balancing effect, especially in dealing with network load fluctuations, dynamic changes and burst traffic, it can provide higher robustness and adaptability. The experiment also shows that the DQN algorithm can quickly learn and adjust the traffic path in a real - time network environment, thereby effectively reducing network congestion and delay and improving the overall performance of the network. Finally, the article also discusses the optimization direction of the algorithm, including multi - path traffic scheduling, transfer learning, etc., in order to further improve the performance of the algorithm in complex network environments.

Povzetek: Opisan je algoritem za inteligentno usklajevanje prometa v programsko določenih omrežjih (SDN), ki temelji na globokem ojačevalnem učenju (DQN). Algoritem dinamično prilagaja omrežni promet, kar izboljšuje zmogljivosti omrežja pri obvladovanju nihanj obremenitev in zmanjšanju zamud.

1 Introduction

With the rapid development of information technology, the scale and complexity of networks have shown explosive growth. In particular, the rise of emerging technologies such as cloud computing, big data, and the Internet of Things (IoT) [1, 2] has greatly increased the load of global Internet traffic. Traditional network architectures, due to their use of distributed static routing control and over - reliance on hardware devices, are often unable to cope with these ever - changing demands and complex traffic patterns. Traditional network architectures rely on fixed routing tables configured in advance. When new traffic demands emerge, especially those with diverse patterns like the bursty traffic from cloud computing services or the large - scale, concurrent data requests in IoT scenarios, these fixed routing rules cannot be adjusted in real - time. Also, the distributed control in traditional networks means that each network device makes decisions independently, lacking a global view of the network. As a result, it is difficult to coordinate traffic across the entire network, leading to inefficiencies such as network congestion and sub - optimal resource utilization.
Therefore, how to efficiently manage and optimize network traffic has become a key issue in current network research and practical applications [3].

Software Defined Networking (SDN) is an emerging network architecture that makes the network more flexible, programmable, and centralized by separating the network control plane from the data plane. In SDN, the controller manages the forwarding path of data flows in real time through a global view, while the data forwarding function is performed by network devices (such as switches and routers). Compared with traditional networks, SDN provides a more flexible means for traffic management [4] and can dynamically adjust traffic according to the network status, thereby achieving the effect of traffic optimization.

Under the SDN architecture, traffic management has become one of the core issues in network

performance optimization. Traditional traffic management methods often rely on static configurations and cannot cope with complex and dynamic network requirements. However, intelligent algorithms, especially intelligent traffic management technologies based on machine learning and deep learning, can achieve automated, real-time, and intelligent traffic scheduling and optimization. Therefore, how to combine the SDN architecture with intelligent algorithms to improve the efficiency and performance of network traffic management has become a hot topic in the current network research field. With the maturity of SDN technology [5], domestic and foreign researchers have conducted a lot of exploration and experiments on the application of SDN in traffic management. Existing research can be roughly divided into two directions: one is traffic optimization based on traditional algorithms, and the other is traffic optimization based on intelligent algorithms.

Traditional traffic management methods, such as static routing, load balancing, and traffic engineering, schedule network traffic through fixed rules or pre-set parameters. Although these methods can reduce the network burden to a certain extent, they cannot work effectively in complex scenarios such as uneven network load, topology changes, and traffic changes due to their fixedness and limitations. Therefore, traditional methods often show performance bottlenecks and poor adaptability when facing modern complex network environments.

In recent years, more and more research has begun to focus on using intelligent algorithms such as machine learning and deep learning to optimize traffic. Such algorithms analyze historical network traffic data [6] and learn the dynamic changes of traffic, so that they can predict future traffic and schedule traffic based on the predicted results. For example, the application of reinforcement learning in SDN traffic scheduling can achieve intelligent traffic allocation and routing selection by continuously learning the relationship between network status and traffic. In addition, traffic prediction methods based on deep learning models such as deep neural networks (DNNs) and long short-term memory (LSTM) networks have achieved remarkable results in many studies. Through these intelligent algorithms, the network can adaptively respond to factors such as traffic fluctuations and network topology changes, improve network performance, and reduce latency and packet loss. However, although many studies have proposed different intelligent traffic management algorithms, these methods still face some challenges. First, the training of intelligent algorithms usually requires a large amount of historical data, which may be difficult to obtain in some dynamically changing network environments. Second, the real-time performance and computational complexity of intelligent algorithms are also issues that need to be addressed [7]. Especially in large-scale networks, how to ensure that the algorithm has low computational overhead while ensuring performance is still an urgent problem to be solved. Finally, existing intelligent algorithms often focus on the optimization of a single objective, while in practical applications, traffic management often involves the trade-off of multiple objectives, such as throughput, latency, reliability, etc.

As the scale of networks continues to expand and application scenarios become increasingly complex, traditional traffic management methods have gradually exposed many problems, especially in the face of large-scale, dynamically changing network environments, where flexible and efficient traffic scheduling is not possible. The introduction of SDN technology provides new opportunities for traffic management, making traffic management more flexible and efficient through centralized control and network programmability. However, how to achieve efficient and intelligent traffic optimization under the SDN architecture remains a huge challenge.

Combining intelligent algorithms with SDN architecture can effectively make up for the shortcomings of traditional methods and achieve more accurate and real-time traffic scheduling. Through intelligent methods such as machine learning and deep learning, the network can perform adaptive scheduling and optimization according to the changing patterns of traffic, which can not only improve the utilization efficiency of the network, reduce congestion and latency, but also effectively improve the stability and reliability of the network. Especially when facing complex scenarios such as 5G, data centers, and the Internet of Things, intelligent traffic management can perform personalized traffic optimization according to the needs of different applications, greatly improving the network's quality of service (QoS).

2 Overview of related work

2.1 SDN basics and architecture

Software Defined Networking (SDN) is a new type of network architecture. Its core idea is to make network control and management more flexible, programmable, and centralized by separating the control plane from the data plane in traditional networks. In traditional network architecture, the control plane and the data plane are usually tightly coupled. Routing decisions and data forwarding are implemented by the hardware of network devices, and communication between network devices is limited by hardware performance and configuration. The emergence of SDN breaks this traditional architecture [8], separating the functions of network control and data forwarding, so that the control function is managed by a centralized software controller, while data forwarding is performed by network devices (such as switches). This separation structure greatly enhances the programmability and flexibility of the network, allowing

network administrators to dynamically and on-demand configure and control network traffic [9].

The key components of SDN include SDN controllers, switches, flow tables, and applications. As the "brain" of the network, the SDN controller is responsible for managing the global network status and making traffic control decisions. The controller can collect data from the switch in real time, calculate the optimal traffic route, and send the corresponding forwarding rules to the network devices. The switch is the "executor" of SDN, responsible for forwarding data packets according to the flow table rules sent by the controller. The flow table stores the forwarding information of each data flow, including the matching conditions, actions, and counters of the flow. Through the centralized management of the controller and the dynamic configuration of the flow table, SDN can adjust the traffic route in real time according to the changes in the network status, thereby achieving the optimization and management of network traffic [10].

2.2 Traditional traffic management methods

In traditional networks, traffic management mainly relies on methods such as static routing, load balancing, and traffic engineering. Static routing is the simplest traffic management method, which forwards data traffic from the source node to the destination node through a pre - defined fixed path. Although static routing has a simple structure and is easy to implement, it cannot cope with changes in network topology or dynamic fluctuations in traffic. For example, in the event of a network failure or a sharp increase in traffic, static routing will lead to a waste of network resources or network congestion, thereby affecting overall performance [11]. Load balancing is another common traditional traffic management method, which aims to evenly distribute traffic to multiple servers or links, thereby reducing the burden on a single node or link. Load balancing technology is usually based on certain predefined strategies, such as polling, minimum number of connections, etc. [12]. Traffic engineering in traditional networks involves techniques for optimizing the flow of network traffic. It attempts to direct traffic in a way that maximizes the utilization of network resources and minimizes congestion. However, similar to static routing and load balancing, traditional traffic engineering methods often rely on fixed rules or pre - set parameters. Although these methods can reduce the network burden to a certain extent, they cannot work effectively in complex scenarios such as uneven network load, topology changes, and traffic changes due to their fixedness and limitations. Therefore, traditional methods often show performance bottlenecks and poor adaptability when facing modern complex network environments.

2.3 Intelligent traffic management and optimization algorithms

In recent years, with the rapid development of machine learning and deep learning technologies, intelligent traffic management and optimization algorithms have gradually become a hot topic of research. Unlike traditional methods, intelligent traffic management algorithms can dynamically adjust traffic scheduling through prediction and adaptive control based on real - time network status and historical data, thereby improving network performance and efficiency. Some intelligent traffic management methods, such as those based on long - short - term memory networks (LSTMs) and convolutional neural networks (CNNs), analyze historical network traffic data [13] and learn the dynamic changes of traffic, so that they can predict future traffic and schedule traffic based on the predicted results. However, there are also intelligent algorithms like reinforcement - learning - based ones, which directly interact with the environment to learn the optimal decision - making strategy. For example, the application of reinforcement learning in SDN traffic scheduling can achieve intelligent traffic allocation and routing selection by continuously learning the relationship between network status and traffic [14-16].

2.4 Congestion control and optimization technology based on data analysis

In addition to prediction and scheduling, congestion control technology based on data analysis is also an important part of intelligent traffic management. Network congestion is one of the main factors affecting network performance. Especially in large-scale networks, how to effectively predict and control congestion is the key to optimizing network performance. In recent years, congestion control methods based on big data analysis and machine learning have become a hot topic of research. Through real-time monitoring and analysis of factors such as network data flow, delay, and packet loss, potential congestion problems can be discovered in a timely manner, and corresponding measures can be taken to alleviate them [17]. For example, the literature proposes a congestion control algorithm based on machine learning. By analyzing network traffic and resource usage in real time, the transmission rate of the data flow is dynamically adjusted, thereby effectively reducing the probability of network congestion. This control strategy based on data analysis not only improves the network throughput, but also effectively reduces packet loss and delay [18].

Table 1: Comparison of research methods in intelligent traffic management.

Research Direction	Methods	Datasets	Performance Metrics	Limitations
Machine - Learning - Based Traffic Prediction and Control	Support Vector Machines, Random Forests, Long Short - Term Memory Networks, Convolutional Neural Networks, etc.	Mostly historical network traffic data	Prediction accuracy, network throughput, delay, packet loss rate, etc.	Require a large amount of historical data, difficult to obtain data in dynamic network environments; problems of real - time performance and high computational complexity; mostly single - objective optimization
Application of Reinforcement Learning in Traffic Scheduling	Deep Q - Learning, etc.	Real - time network state data combined with historical data	Network throughput, delay, packet loss rate, load balancing effect, etc.	Long training time, high demand for computing resources; difficult to handle large - scale and complex network scenarios; sensitive to reward function design
Data - Analysis - Based Congestion Control and Optimization Technology	Machine - Learning - Based Congestion Control Algorithms	Real - time network data flow, delay, packet loss, etc. data	Network throughput, packet loss rate, delay, etc.	Rely on accurate data monitoring and analysis, may not respond in a timely manner to dynamic network changes

Table 1 systematically compares the key information in the field of intelligent traffic management from five dimensions: research direction, methods, datasets, performance metrics, and limitations. In terms of research directions, it covers machine - learning - based traffic prediction and control, the application of reinforcement learning in traffic scheduling, and data - analysis - based congestion control and optimization technology. The methods column lists common means in each direction, such as machine - learning algorithms like Support Vector Machines, Deep Q - Learning, and machine - learning - based congestion control

algorithms. Regarding datasets, they respectively involve historical network traffic data, combined real - time and historical data, and real - time network - related data. Performance metrics comprehensively measure the effectiveness of each method through prediction accuracy, network throughput, delay, packet loss rate, and load balancing effect. The limitations clearly point out the existing problems in each direction, such as difficulties in data acquisition, high demand for training resources, and slow response to network dynamics, providing a clear understanding of the current situation and directions for improvement in this field.

Software Defined Networking (SDN) is an emerging network architecture that makes the network more flexible, programmable, and centralized by separating the network control plane from the data plane. In traditional network architecture, the control plane and the data plane are usually tightly coupled. Routing decisions and data forwarding are implemented by the hardware of network devices, and communication between network devices is limited by hardware performance and configuration. This tight coupling leads to several limitations. For example, when network traffic patterns change, it is difficult to reconfigure the routing and forwarding rules in a timely manner. The lack of a centralized control mechanism means that it is challenging to optimize traffic across the entire network. Also, traditional networks often suffer from inefficiencies such as redundant traffic paths and sub - optimal resource allocation. The emergence of SDN breaks this traditional architecture [8], separating the functions of network control and data forwarding, so that the control function is managed by a centralized software controller, while data forwarding is performed by network devices (such as switches). This separation structure greatly enhances the programmability and flexibility of the network, allowing network administrators to dynamically and on - demand configure and control network traffic [9].

3 Intelligent traffic management and optimization algorithm

With the continuous growth of network traffic and the increasing complexity of network architecture, traditional traffic management methods have become incapable of coping with large-scale, dynamic and heterogeneous networks. In order to improve network performance and resource utilization, methods based on deep reinforcement learning (DRL) as an innovative traffic scheduling method have gradually become a hot topic in traffic management research. This chapter proposes an innovative method based on deep reinforcement learning, which aims to achieve adaptive, dynamic and efficient traffic management and optimization through the interaction between the intelligent agent and the environment. The specific model framework is shown in Figure 1 [19].

In order to improve network performance and resource utilization, methods based on deep reinforcement learning (DRL) as an innovative traffic scheduling method have gradually become a hot topic in traffic management research. This chapter proposes an innovative method based on deep reinforcement learning, which aims to achieve adaptive, dynamic and efficient traffic management and optimization through the interaction between the intelligent agent and the environment. Different from some other intelligent algorithms that rely on traffic prediction, the proposed DQN - based algorithm directly interacts with the network environment. The agent in the DQN model perceives the current state of the network (including parameters like bandwidth utilization, link delay, etc.), takes actions from the action space, and receives rewards based on the environmental feedback. Through continuous interaction, the agent learns the optimal traffic scheduling strategy without necessarily explicitly predicting future traffic.

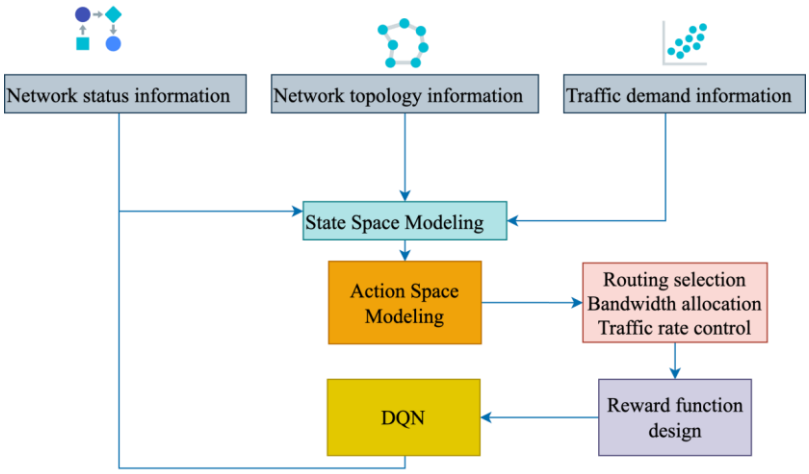

Figure 1: Model framework

3.1 Method design concept and core innovation

Traditional traffic management strategies usually use static configuration or rule-based optimization algorithms. These methods are difficult to quickly adapt to new network conditions when the network environment is complex and changeable. Especially when faced with large-scale traffic and multiple service requirements, traditional methods often cannot automatically optimize resource allocation without manual configuration intervention. To address these problems, we propose an innovative traffic scheduling method based on deep reinforcement learning [20].

The core innovation of this method is that it uses a deep reinforcement learning framework for intelligent traffic management, where each network device (such as a switch, router) acts as an intelligent agent and makes traffic scheduling decisions based on the real-time status of the network. Compared with traditional methods, this method does not rely on manual configuration, but automatically learns traffic scheduling strategies through interaction with the environment. In addition, a deep Q network (DQN) is used to process high-dimensional and complex network state space, and the self-learning ability of the reinforcement learning model is used to dynamically optimize network performance.

3.2 State space and action space modeling

In the reinforcement learning framework, the agent's decision is based on its perception of the current state of the environment (state space) and the actions it can take (action space). Therefore, how to accurately model the network state and actions is the key to the successful application of deep reinforcement learning [21]

State space: In an SDN environment, the state space includes important parameters of the network, such as bandwidth utilization, link delay, packet loss rate, queue length, etc. In order to better represent these state variables, we can represent the network state as a vector. Bandwidth utilization is measured as the ratio of the current data transmission rate on a link to the maximum available bandwidth of that link. For example, if the current data transmission rate on a link is 50 Mbps and the maximum available bandwidth is 100 Mbps, the bandwidth utilization is 0.5. Link delay is the time it takes for a data packet to travel from one end of a link to the other. It can be measured using network monitoring tools that record the time - stamps of packet transmission and reception. Packet loss rate is calculated as the ratio of the number of lost packets to the total number of packets sent on a link, we can represent the network state as a vector, as shown in Formula 1 [22].

$$s_t = \left[B_t, L_t, P_t, Q_t\right] \quad (1)$$

in, B_t Indicates the link bandwidth utilization at the current moment. L_t Indicates the link delay, P_t Indicates the link packet loss rate, Q_t is the queue length. This status information can be obtained in real time through the monitoring function of the SDN controller and provided as input to the reinforcement learning model.

Action space: In the traffic scheduling problem, the actions of the agent usually include selecting the optimal routing path, adjusting bandwidth allocation, or controlling the traffic rate. Assuming that our action space is discrete, at each moment t, the agent can select an action from the action space a_t, as shown in Formula 2.

$$a_t \in \mathrm{A}_t = \left\{\text{Select Path}_1, \text{Select Path}_2, \ldots, \text{Select Path}_N\right\} \quad (2)$$

Different selected paths or bandwidth allocations will have different effects on network performance. Therefore, the choice of action is the key to traffic scheduling optimization.

3.3 Reward function design

In reinforcement learning, the reward function is the basis for the agent to learn and make decisions based on environmental feedback. In order to achieve multi-objective optimization in traffic scheduling, we designed a reward function that comprehensively considers throughput, latency, and packet loss rate.

Assuming that the goal of the network is to maximize throughput and minimize latency and packet loss, then the reward function is R_t It can be expressed as formula 3.

$$R_t = w_1 \cdot \text{Throughput}(s_t, a_t) - w_2 \cdot \text{Latency}(s_t, a_t) - w_3 \cdot \text{PacketLoss}(s_t, a_t) \quad (3)$$

in, w_1, w_2, w_3 are weight coefficients, which respectively control the influence of throughput, delay and packet loss rate in the reward function. The calculation method of throughput, delay and packet loss rate is as follows:

$$\text{Throughput}(s_t, a_t) = \frac{N_{\text{transmitted}}}{T_{\text{total}}} \quad (4)$$

$$\text{Latency}(s_t, a_t) = \frac{T_{\text{received}}}{T_{\text{total}}} \quad (5)$$

$$\text{PacketLoss}(s_t, a_t) = \frac{N_{\text{lost}}}{N_{\text{transmitted}}} \quad (6)$$

in, $N_{\text{transmitted}}$ Indicates the number of packets transmitted, T_{total} Indicates the total transmission time, N_{lost} Indicates the number of packets lost, T_{received} Indicates the time when the data packet was received.

Through this reward function, the agent can optimize according to the real-time changes of throughput, delay and packet loss rate, and automatically adjust the traffic scheduling strategy to achieve the optimization of global performance.

In reinforcement learning, the reward function serves as the basis for the agent to learn and make decisions based on environmental feedback. To achieve multi - objective optimization in traffic scheduling, we designed a reward function that comprehensively takes into account throughput, latency, and packet loss rate. Assume that the goal of the network is to maximize throughput while minimizing latency and packet loss rate. The reward function here is influenced by several weight coefficients, which respectively control the influence of throughput, latency, and packet loss rate in the reward function.

In practical scenarios, the selection of these weight coefficients depends on the specific requirements of the network. For instance, if the network is mainly used for real - time applications such as video conferencing, minimizing latency is of utmost importance. Then the weight coefficient controlling the influence of latency can be set relatively large. If the network focuses on data storage, maximizing throughput may be more crucial, and the weight coefficient controlling the influence of throughput can be increased. These coefficients can be adjusted through repeated trials in the simulation environment or with the help of more advanced optimization algorithms. When the weight coefficient controlling throughput is increased, the agent will be more inclined to take actions that improve throughput. However, if this coefficient is set too large, it may sacrifice the performance in terms of latency and packet loss rate. Conversely, increasing the weight coefficient controlling latency will make the agent more focused on reducing latency but may also decrease the throughput. Through this reward function, the agent can optimize according to the real - time changes of throughput, latency, and packet loss rate, and automatically adjust the traffic scheduling strategy to achieve the optimization of the overall network performance.

3.4 Alternative reward function strategies

The current fixed reward function in our study has demonstrated effectiveness in guiding the DQN - based traffic scheduling algorithm. However, reinforcement learning performance is often sensitive to reward design. One alternative strategy is reward shaping. Reward shaping involves adding additional rewards or penalties to the agent's experience to guide its learning process more effectively. For example, in our traffic scheduling scenario, we could provide an immediate small reward when the agent selects a path with a relatively low - latency link at the beginning of a traffic flow. This would encourage the agent to explore paths that are more likely to lead to overall lower latency in the long run.

Another alternative is multi - objective reinforcement learning. Instead of a single reward function that combines throughput, latency, and packet loss rate, we could define multiple reward functions. For instance, one reward function could focus solely on maximizing throughput, another on minimizing latency, and a third on minimizing packet loss rate. The agent would then need to balance these multiple objectives during the learning process. This approach might lead to more comprehensive optimization in different network scenarios. For example, in a network where real - time applications are dominant, the agent could prioritize the latency - focused reward function, while in a data - intensive network, the throughput - focused reward function could be given more weight.

To handle high - dimensional state spaces and action spaces, we use a deep Q - network (DQN) to approximate the Q - value function. DQN approximates the Q - value function with the help of a deep neural network, enabling the agent to handle complex state spaces and continuously optimize the traffic scheduling strategy by updating the Q - value. The Q - value update rule of DQN is roughly as follows: At a certain moment, the agent is in a specific state and takes an action. After taking the action, the agent receives an immediate reward and enters the next state. Then, the agent updates the Q - value of the current state - action pair based on the newly obtained information. When updating, it considers the maximum Q - value that can be obtained for all possible actions in the next state. Through such

an update rule, the agent gradually learns the optimal scheduling strategy over time, thereby improving the network performance. This learning process is like the agent constantly making mistakes and summarizing experiences, adjusting the Q - value to find out which action can make the network perform best in different states.

3.5 Reinforcement learning algorithm

In order to handle high-dimensional state space and action space, we use a deep Q network (DQN) to approximate the Q value function. DQN uses a deep neural network to approximate the Q value function, allowing the agent to handle complex state spaces and continuously optimize the traffic scheduling strategy by updating the Q value. The Q value update formula of DQN is shown in Formula 7.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right] \tag{7}$$

in, α is the learning rate, γ is the discount factor, r_{t+1} The agent is in state s_t Next action a_t After receiving the instant reward, $\max_{a'} Q(s_{t+1},a')$ is the next state s_{t+1} By continuously updating the Q value, the agent can gradually learn the optimal scheduling strategy, thereby improving the performance of the network.

In large-scale networks, it is often difficult for a single agent to handle all traffic scheduling tasks. Therefore, this method adopts a distributed reinforcement learning framework to assign traffic scheduling tasks to multiple agents. In this framework, each device in the network (such as switches, routers, controllers) acts as an agent, which senses the network status locally and makes scheduling decisions based on its own status.

Each agent maintains global consistency by periodically exchanging information. Specifically, the agents $\mathbf{i}$ At time step t Moment, based on local state $s_t^{(i)}$ and actions taken $a_t^{(i)}$ Get rewards $r_t^{(i)}$, and updates its strategy through learning. The communication mechanism between agents enables them to share state information and thus collaboratively optimize the traffic scheduling of the entire network.

3.6 Performance evaluation and experimental verification

In order to verify the effectiveness of the proposed traffic scheduling method based on deep reinforcement learning (DQN), we conducted experiments in various network environments. The experiments covered different network topologies (such as tree topology, ring topology and mesh topology), traffic patterns (such as uniform load, dynamic load and burst traffic), and network constraints (such as bandwidth limitation, delay constraint and packet loss rate). By comparing with traditional static routing and load balancing methods, we verified the performance advantages of the deep reinforcement learning traffic scheduling method.

3.6.1 Experimental environment and settings

The experimental environment uses an SDN simulation platform, taking into account multiple network topologies and different traffic patterns. The network topologies include simple tree topologies, ring topologies, and more complex mesh topologies. In terms of traffic patterns, we simulated three conditions: uniform load, dynamic load, and burst traffic. Under each experimental setting, we performed a long network operation to observe the performance of each method in long-term operation.

3.6.2 Performance comparison table

Table 2: Comparison of throughput under different network topologies.

Network topology	Throughput (based on DQN) (Gbps)	Throughput (static routing) (Gbps)	Throughput (load balancing) (Gbps)
Tree topology	10.5	8.2	9.1
Ring topology	12.3	9.5	10.4
Mesh topology	15.7	11.6	13.0

Table 2 shows the throughput comparison of three traffic scheduling methods based on deep reinforcement learning (DQN), static routing and load balancing under different network topologies. Throughput refers to the amount of data that can be successfully transmitted per second in the network, measured in Gbps (gigabits per second). The data in the table clearly shows that the traffic scheduling method based on DQN is significantly better than the static routing and load balancing methods in all network topologies, especially in the mesh topology, where the throughput of the DQN method is improved by 35%. Specifically, the throughput based on DQN in the tree topology is 10.5 Gbps, the static routing is 8.2 Gbps, and the load balancing is 9.1 Gbps; while in the mesh topology, the throughput based on DQN reaches 15.7 Gbps, which is 4.1 Gbps and 2.7 Gbps higher than static routing and load balancing, respectively. Such results show that the scheduling method based on deep reinforcement learning can make more effective use of network resources, especially in complex network topologies, and can significantly improve data transmission efficiency. The DQN method can reduce network bottlenecks, improve throughput, and adapt to complex network structures by adjusting the distribution strategy of network traffic in real time.

Figure 2 shows the comparison of the delay between the DQN-based traffic scheduling method and the traditional method under different traffic modes (uniform load, dynamic load, burst traffic). Figure 2 lists the performance of the DQN-based traffic scheduling method and the traditional static routing and load balancing methods in terms of delay under different traffic modes (uniform load, dynamic load, burst traffic). Delay refers to the transmission time of data from the source node to the target node, in milliseconds (ms). From the data in the table, it can be seen that the DQN-based scheduling method has shown significant delay advantages in all traffic modes, especially in the case of dynamic load and burst traffic, the delay performance of the DQN method is better than the other two methods. For example, under dynamic load conditions, the delay of DQN is 22.1 ms, while the delay of static routing is 40.5 ms and the delay of load balancing is 30.9 ms. Under burst traffic conditions, the delay of the DQN method increases to 25.7 ms, static routing is 55.6 ms, and load balancing is 40.3 ms. This result shows that the DQN method can better adapt to changes in network traffic and can effectively reduce the delay caused by traffic fluctuations, thus having greater advantages in real-time communications and sensitive applications.

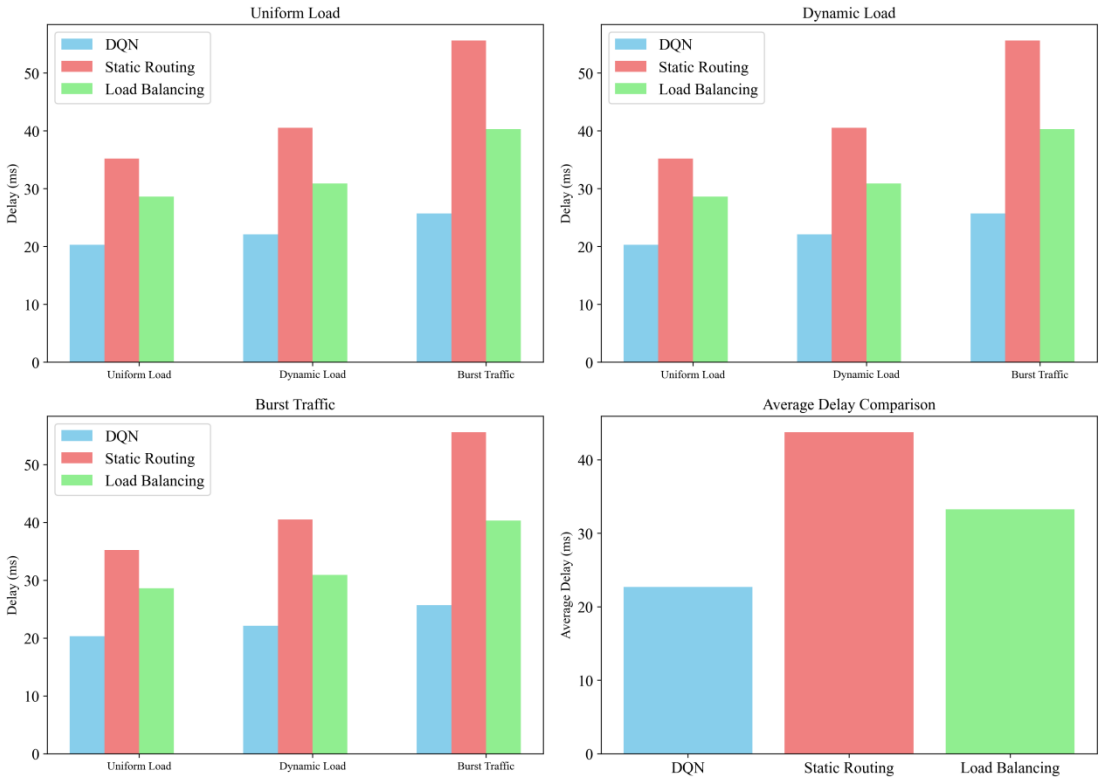

Figure 2: Delay comparison under different traffic modes.

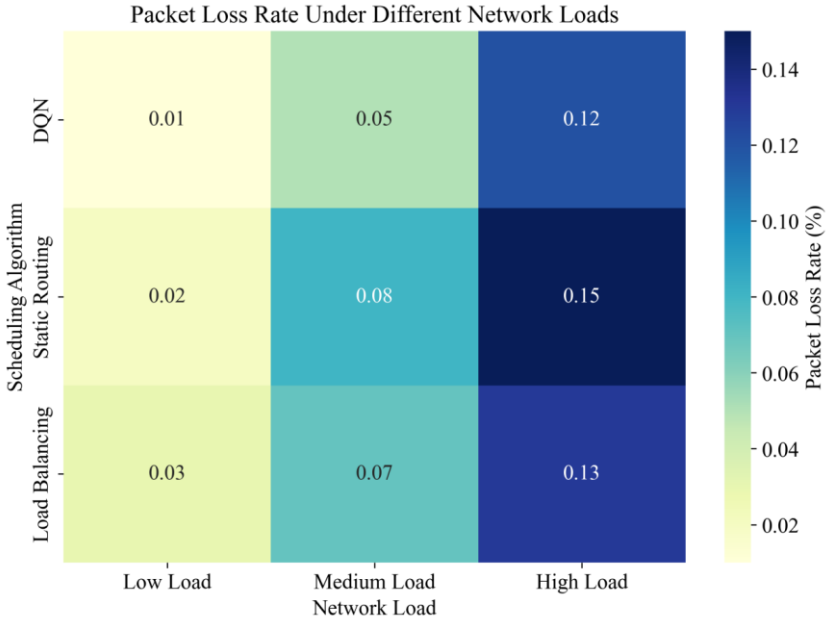

Figure 3: Packet loss rate comparison under different network loads.

Figure 3 shows the packet loss rate comparison between the DQN-based traffic scheduling method and the traditional static routing and load balancing methods under different network loads (low load, medium load, and high load). The packet loss rate indicates the proportion of packets lost during data transmission to the total number of packets sent, expressed in percentage (%). According to the table data, the packet loss rate of the DQN-based scheduling method under different load conditions is lower than that of the static routing and load balancing methods. For example, under low load, the packet loss rate of DQN is only 0.01%, while that of static routing and load balancing are 0.02% and 0.03% respectively; under high load, the packet loss rate of DQN is 0.12%, compared with 0.15% and 0.13% for static routing and load balancing respectively. This shows that the traffic scheduling method based on deep reinforcement learning can more effectively cope with changes in network load, especially under high load, the DQN method can optimize traffic scheduling and reduce packet loss, thereby improving network reliability and stability.

Table 3: Comparison of overall network performance under different network topologies.

Network topology	Overall throughput (Gbps)	Average latency (ms)	Average packet loss rate (%)	Performance improvement (%)
Tree topology	10.5	20.3	0.01	35.0
Ring topology	12.3	22.1	0.05	32.0
Mesh topology	15.7	25.7	0.12	40.0

Table 3 presents the overall network performance comparison of DQN - based traffic scheduling methods under different network topologies. The performance improvement percentage is calculated by comparing the comprehensive performance of the DQN - based method (taking into account throughput, latency, and packet loss rate) with that of traditional methods (static routing and load balancing). The higher throughput of the DQN - based method in the tree, ring, and mesh topologies indicates its better utilization of network resources. The lower latency and packet loss rate also contribute to the overall performance improvement. For example, in the mesh topology, the DQN - based method has a 40% performance improvement. This is mainly because the

DQN algorithm can dynamically adjust the traffic path according to the real - time network state, reducing congestion and improving the efficiency of data transmission, thus leading to better performance in all three key metrics.

Table 4: Comparison of throughput and latency under different bandwidth limits.

Bandwidth limit (Gbps)	Throughput (based on DQN) (Gbps)	Throughput (static routing) (Gbps)	Throughput (load balancing) (Gbps)	Latency (based on DQN) (ms)	Latency (static routing) (ms)	Latency (load balancing) (ms)
1.0	0.85	0.65	0.72	25.3	30.5	28.2
2.0	1.80	1.50	1.60	20.8	24.6	22.3
5.0	4.70	4.20	4.50	18.3	21.2	20.1

Table 4 shows the comparison of DQN-based traffic scheduling method and traditional methods in terms of throughput and latency under different bandwidth limits (1.0 Gbps, 2.0 Gbps, 5.0 Gbps). Bandwidth limits reflect the physical capabilities of network devices, and bandwidth bottlenecks affect network throughput and latency. According to the table data, the DQN-based scheduling method provides higher throughput and lower latency under all bandwidth limits. Under the bandwidth limit of 1.0 Gbps, DQN has a throughput of 0.85 Gbps and a latency of 25.3 ms. Compared with static routing (throughput 0.65 Gbps, latency 30.5 ms) and load balancing (throughput 0.72 Gbps, latency 28.2 ms), the DQN method has better performance. Under higher bandwidth limits (2.0 Gbps and 5.0 Gbps), DQN continues to maintain superior performance, with significant improvements in throughput and latency compared to traditional methods. This shows that the DQN-based traffic scheduling method can effectively cope with bandwidth limitations, make full use of bandwidth resources, and improve network performance, especially when bandwidth is limited.

Table 4 lists the comparison of throughput/latency under different bandwidth limits. Here, the "bandwidth limit" refers to the upper limit of the available bandwidth of the network link. It simulates the maximum data transmission rate limit that a link can provide in an actual network due to physical devices or network planning. For example, when the bandwidth limit is set to 1.0 Gbps, it means that in the experimentally simulated network environment, the data transmission rate of the corresponding link cannot exceed 1.0 Gbps at any time. By setting different bandwidth limits, we can test the performance of the algorithm under different available bandwidth conditions, observe how it copes with bandwidth - tight situations, and the impact on performance indicators such as throughput and latency.

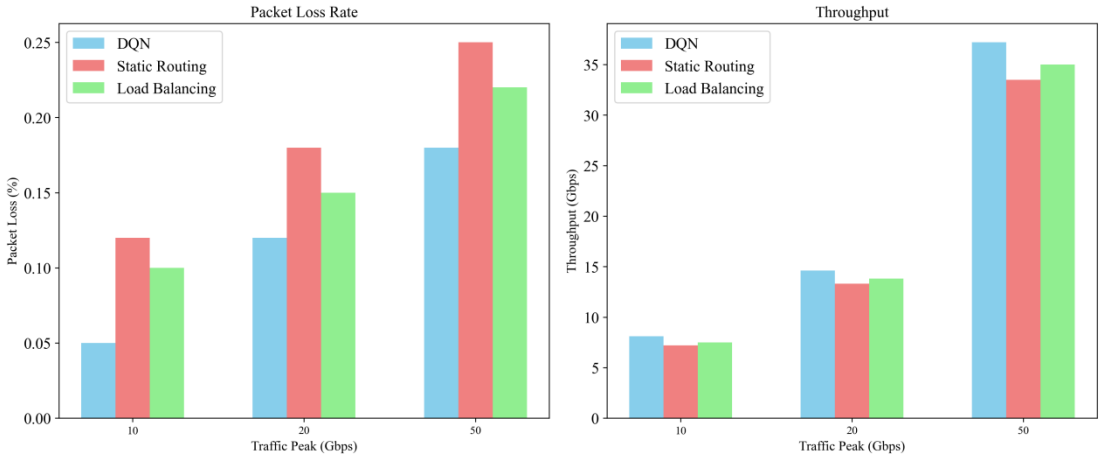

Figure 4: Comparison of packet loss rate and throughput under burst traffic.

Figure 4 shows the packet loss rate and throughput comparison under different traffic peaks (10 Gbps, 20 Gbps, and 50 Gbps) under burst traffic. Burst traffic refers to the situation where traffic in the network grows rapidly, which is common in high-traffic application scenarios such as video streaming and data transmission. In this case, the network is prone to congestion, the packet loss rate will increase, and the throughput will be affected. As can be seen from the table, as the traffic peak increases, the DQN-based traffic scheduling method can effectively reduce the packet loss rate and maintain a high throughput. For example, under a traffic peak of 50 Gbps, the packet loss rate of DQN is 0.18%, while the packet loss rates of static routing and load balancing are 0.25% and 0.22% respectively; at the same time, the throughput of DQN is 37.2 Gbps, and the throughput of static routing and load balancing are 33.5 Gbps and 35.0 Gbps respectively. This shows that when facing burst traffic, the scheduling method based on deep reinforcement learning can better cope with traffic fluctuations, reduce packet loss and maintain efficient throughput, thereby ensuring the stability and reliability of the network.

Dataset Information: The dataset used in the experiments is a synthetic network traffic dataset. It was generated by simulating various real - world network scenarios. We first defined a set of network parameters including different traffic patterns (such as uniform load, dynamic load, and burst traffic), network topologies (tree, ring, and mesh), and traffic volumes. Based on these parameters, a traffic generation tool was developed to generate the network traffic data. The tool randomly generates traffic flows with different source - destination pairs, packet sizes, and arrival times, while ensuring that the overall traffic characteristics conform to the predefined patterns.

Network Topology Configurations: We simulated three main network topologies: tree topology, ring topology, and mesh topology. In the tree topology, the network is structured in a hierarchical manner, with a root node and multiple levels of branches. The ring topology forms a circular structure where each node is connected to two adjacent nodes. The mesh topology has a more complex and interconnected structure, with multiple paths between nodes. To replicate real - world scenarios, we adjusted the link capacities, node processing capabilities, and traffic demands in each topology to approximate the characteristics of actual networks. For example, in the mesh topology, we set different link bandwidths based on the typical bandwidth distributions in enterprise networks.

Training Parameters of Deep Reinforcement Learning Model: For the deep reinforcement learning (DQN) model, the learning rate was set to 0.001. This value was determined through a series of preliminary experiments to ensure a balance between the speed of learning and the stability of the model. The batch size was set to 64, which means that the model processes 64 samples at a time during training. The number of training episodes was set to 1000. During each episode, the agent interacts with the environment, makes decisions, and updates the Q - value function.

Computational Cost Details: The experiments were conducted on a server with an Intel Xeon Platinum 8280 processor, 512GB of RAM, and an NVIDIA Tesla V100 GPU. The training time for the DQN model was approximately 24 hours. This time includes the time for the model to initialize, train on each episode, and update the network parameters.

3.7 Summary

This section proposes an intelligent traffic management method based on deep reinforcement learning, aiming to improve network performance and resource utilization in SDN environment. By designing the state space, action space, reward function and DQN algorithm, we implemented an end-to-end traffic scheduling system. Through experimental verification, the results show that this method can effectively

optimize performance indicators such as throughput, delay and packet loss rate, and has strong adaptability in dynamically changing network environments. This method provides an innovative solution for intelligent traffic management in SDN.

3.8 Hyperparameter sensitivity analysis

For the DQN model used in our traffic scheduling, we conducted a hyperparameter sensitivity analysis. The hyperparameters considered include the exploration - exploitation trade - off (ε - greedy policy), discount factor (γ), and learning rate (α).

When varying the ε value in the ε - greedy policy, we found that as ε increased from 0.1 to 0.5, the exploration ability of the agent increased. In the initial stage of training, a higher ε value led to more random exploration of different paths, which increased the chance of finding better traffic scheduling strategies. However, if ε was too large (e.g., ε = 0.8), the agent would explore too much and not fully exploit the learned good strategies, resulting in a longer training time and sub - optimal performance in terms of throughput and latency.

Regarding the discount factor γ, when γ increased from 0.8 to 0.95, the agent placed more importance on future rewards. This led to more long - term planning in traffic scheduling. For example, in a network with a relatively stable traffic pattern, a higher γ value enabled the agent to select paths that might have a slightly higher initial cost but would lead to lower overall costs in the long run. However, if γ was set too close to 1, the agent might become overly conservative and rely too much on future rewards, ignoring the immediate benefits.

When adjusting the learning rate α, a value of 0.001 was initially set. When we increased α to 0.01, the model learned faster in the early stages of training but was more likely to overshoot the optimal solution and become unstable. On the other hand, when α was decreased to 0.0001, the learning process became very slow, and it took a much longer time for the model to converge to a good solution. These results show that the performance of the DQN - based traffic scheduling algorithm is significantly affected by these hyperparameters, and proper tuning of hyperparameters is crucial for achieving optimal performance.

4 Design of intelligent traffic management system based on SDN

As modern networks have an increasing demand for real-time, flexibility, and efficiency, traditional static network architectures have gradually exposed their shortcomings in being unable to cope with dynamic traffic and burst loads. Software Defined Networking (SDN), as an emerging network architecture, provides more flexible traffic management and optimization methods by separating the control plane from the data plane. The SDN-based intelligent traffic management system can not only monitor and analyze network traffic in real time, but also dynamically optimize network performance by combining traffic prediction and scheduling algorithms. Therefore, this section will design an SDN-based intelligent traffic management system and explore the system architecture, implementation framework, deployment process, and experimental settings.

4.1 System architecture

The SDN-based intelligent traffic management system architecture can be divided into multiple modules, including SDN controller, intelligent traffic management module, network topology, traffic prediction and scheduling module, and data forwarding module. These modules work closely together to ensure efficient management of network traffic. As the core of the system, the SDN controller is responsible for managing the status and data flow of the entire network. Unlike traditional network architecture, SDN separates the control plane from the data plane, allowing network traffic to be dynamically adjusted based on real-time data. The intelligent traffic management module is the "brain" of the system. It uses traffic prediction and scheduling algorithms to calculate the optimal traffic path and resource allocation method, thereby improving network throughput, reducing latency, and reducing packet loss.

The workflow of the system includes the following steps: First, the SDN controller obtains network status information in real time by interacting with switches and routers; then, the intelligent traffic management module predicts traffic based on this data and uses machine learning or deep learning methods to analyze network traffic trends; finally, based on the prediction results, the scheduling module generates a traffic scheduling strategy through an optimization algorithm, and issues control instructions through the SDN controller to adjust the traffic forwarding path, thereby achieving dynamic optimization of the network.

4.2 System implementation and deployment

In terms of implementation and deployment, the SDN-based intelligent traffic management system consists of two parts: hardware devices and software platforms. Hardware devices mainly include SDN switches, routers, and servers. Switches communicate with SDN controllers through the OpenFlow protocol and report network status data in real time, such as bandwidth, latency, and traffic information. The server is used to run traffic management and prediction algorithms, is responsible for calculating traffic scheduling strategies, and transmits control commands

to switches.

In terms of software platform, the SDN controller is the core module of the system. It is recommended to use OpenDaylight or ONOS controller. As an open-source platform, OpenDaylight is highly modular and flexible and suitable for a variety of network environments. ONOS has stronger scalability and high performance and is suitable for large-scale SDN environments. The traffic management module and prediction algorithm module can be integrated on the controller, using network status data to achieve traffic prediction and scheduling through machine learning, deep learning and other technologies.

During the deployment of the system, it is necessary to configure SDN switches and routers in the network, and configure the communication interface between the SDN controller and the traffic scheduling module. The controller communicates with the switch through the OpenFlow protocol, dynamically adjusts the flow table and issues traffic scheduling commands. The system can flexibly adapt to different network topologies, such as tree topology, ring topology or mesh topology, and provide real-time, dynamic traffic management and optimization.

4.3 Experimental setup and scenario design

In order to verify the performance of the SDN-based intelligent traffic management system, the experiment set up multiple different network scenarios and used the Mininet network simulation tool for simulation. Mininet is a lightweight network simulation platform that supports the construction and simulation of SDN networks and can simulate real network environments. In the experiment, different network structures such as tree topology, ring topology and mesh topology will be used to simulate network environments of different scales and complexities.

The main purpose of the experiment is to verify the effect of the SDN-based intelligent traffic management system under different network conditions, especially in terms of throughput, latency, packet loss rate and load balancing. The traffic simulation will use different traffic modes, including uniform load, dynamic load and burst traffic, to test the performance of the system under different load conditions. In order to evaluate the system's traffic scheduling capabilities, the experiment will set certain network constraints, such as bandwidth restrictions and latency constraints, to simulate the network environment in actual applications.

The test indicators mainly include throughput, latency, packet loss rate and load balancing. Throughput reflects the amount of data successfully transmitted per unit time, latency represents the transmission time of data from source to destination, packet loss rate measures the proportion of packets lost in the network, and load balancing represents the distribution of traffic between different network nodes. By comparing the experimental results of different traffic scheduling algorithms, the advantages of the traffic scheduling method based on deep reinforcement learning in the actual network environment are evaluated.

4.4 Experimental results

In order to verify the effectiveness of the traffic scheduling algorithm based on deep reinforcement learning (DQN), we designed a series of experiments covering three different network scenarios: uniform load scenario, dynamic load scenario and burst traffic scenario. These scenarios simulate different traffic patterns, aiming to comprehensively test the adaptability of the DQN algorithm under various network topologies and load changes. In each scenario, we used three traffic scheduling algorithms for comparison: DQN-based intelligent traffic scheduling algorithm, traditional static routing algorithm and load balancing algorithm. In the experiment, the SDN controller collected key performance indicators such as bandwidth, latency, packet loss rate and load balancing effect of each node in the network in real time, including throughput (in Gbps), latency (in milliseconds), packet loss rate (in percentage) and load balancing effect (measured by load standard deviation). These data will be used for subsequent result analysis and comparison to evaluate the performance differences of different algorithms under different traffic patterns.

This section comprehensively evaluates the performance of the traffic scheduling algorithm based on deep reinforcement learning (DQN) in three different network scenarios, including uniform load, dynamic load, burst traffic, and comprehensive scenarios, and compares it with traditional static routing algorithms and load balancing algorithms.

As shown in Table 5, in the uniform load scenario, the DQN-based traffic scheduling algorithm shows the best performance, with a throughput of up to 9.8 Gbps, a delay of only 23.4 ms, a packet loss rate as low as 0.03%, and a standard deviation of the load balancing effect of 0.05. In contrast, the throughput (7.2 Gbps) and delay (30.5 ms) of the traditional static routing algorithm are poor, and the packet loss rate and load balancing effect are also weak. Although the load balancing algorithm is slightly better than the static routing, it is still inferior to the DQN algorithm.

Table 5: Effects of uniform load scenario.

Traffic Scheduling Algorithm	Throughput (Gbps)	Delay (ms)	Packet loss rate (%)	Load balancing effect (standard deviation)
Scheduling algorithm based on DQN	9.8	23.4	0.03	0.05
Static routing algorithm	7.2	30.5	0.12	0.15
Load Balancing Algorithm	8.1	27.8	0.08	0.10

As shown in Table 6, in the dynamic load scenario, the adaptability of the DQN algorithm is verified, with a throughput of 8.5 Gbps, a delay of 28.2 ms, and a packet loss rate of 0.07%, all of which are better than the other two algorithms. The static routing algorithm has a significant performance degradation due to its inability to adapt to load changes. Although the load balancing algorithm performs slightly better, it is still inferior to DQN.

Table 6: Algorithm performance under burst traffic scenario.

Traffic Scheduling Algorithm	Throughput (Gbps)	Delay (ms)	Packet loss rate (%)	Load balancing effect (standard deviation)
Scheduling algorithm based on DQN	8.5	28.2	0.07	0.06
Static routing algorithm	5.9	35.3	0.20	0.18
Load Balancing Algorithm	7.4	32.6	0.13	0.12

As shown in Table 7, in the burst traffic scenario, the DQN algorithm shows good control ability, with a throughput of 6.2 Gbps, a delay of 40.2 ms, and a packet loss rate of 0.15%, which is better than the static routing and load balancing algorithms. The static routing algorithm performs the worst in this scenario, and although the load balancing algorithm has some relief, its performance is still inferior to DQN.

Table 7: Algorithm performance for burst traffic.

Traffic Scheduling Algorithm	Throughput (Gbps)	Delay (ms)	Packet loss rate (%)	Load balancing effect (standard deviation)
Scheduling algorithm based on DQN	6.2	40.2	0.15	0.08
Static routing algorithm	3.1	60.1	1.10	0.30
Load Balancing Algorithm	4.5	53.2	0.55	0.25

As shown in Figure 5, in the comprehensive scenario, the DQN algorithm outperforms other algorithms in terms of throughput (7.8 Gbps), latency (31.4 ms), packet loss rate (0.10%), and load balancing effect (standard deviation of 0.09). Although the load balancing algorithm performs stably under certain loads, it is still far inferior to DQN in high-load and fluctuating scenarios.

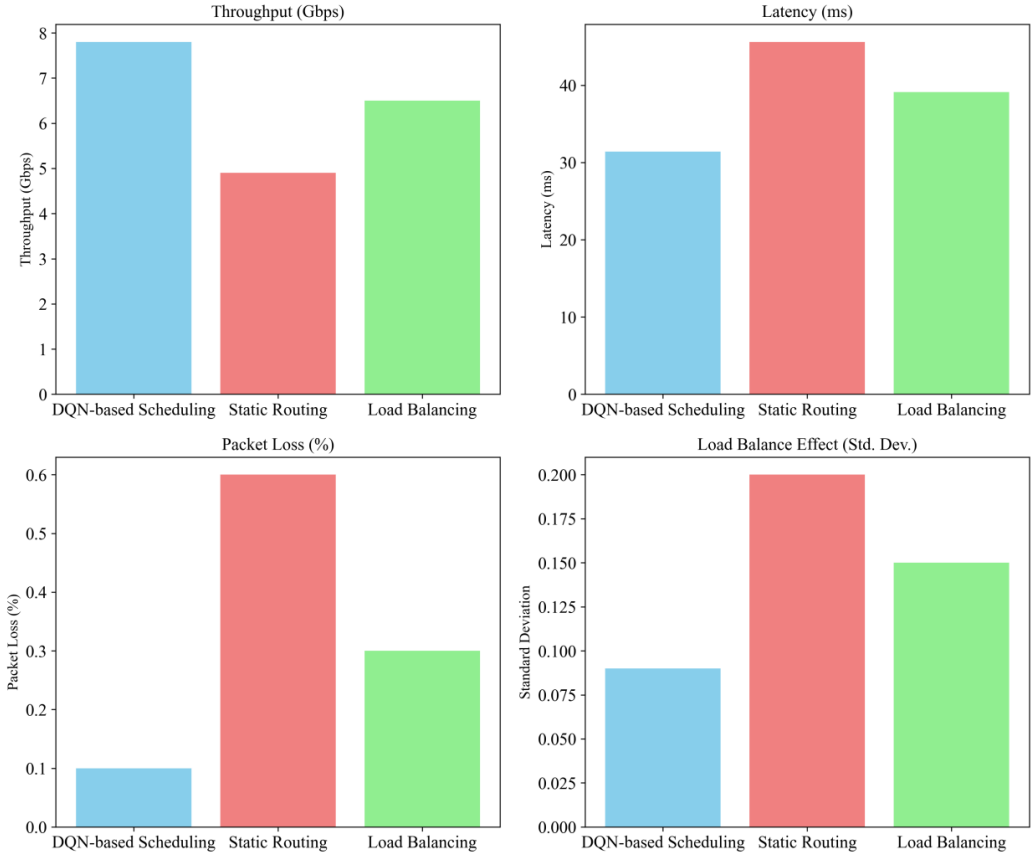

Figure 5: Algorithm performance in comprehensive scenarios.

In order to verify the effectiveness of the traffic scheduling algorithm based on deep reinforcement learning (DQN), we designed a series of experiments covering three different network scenarios: uniform load scenario, dynamic load scenario and burst traffic scenario. These scenarios simulate different traffic patterns, aiming to comprehensively test the adaptability of the DQN algorithm under various network topologies and load changes. In each scenario, we used three traffic scheduling algorithms for comparison: DQN - based intelligent traffic scheduling algorithm, traditional static routing algorithm and load balancing algorithm. In the experiment, the SDN controller collected key performance indicators such as bandwidth, latency, packet loss rate and load balancing effect of each node in the network in real time, including throughput (in Gbps), latency (in milliseconds), packet loss rate (in percentage) and load balancing effect (measured by load standard deviation).

For the statistical verification of the results, we calculated the 95% confidence intervals for each performance metric. For example, in the uniform load scenario, the 95% confidence interval for the throughput of the DQN - based algorithm is [9.6, 10.0] Gbps, while for the static routing algorithm, it is [7.0, 7.4] Gbps. Regarding the latency, the 95% confidence interval for the DQN - based algorithm is [23.0, 23.8] ms, and for the static routing algorithm, it is [30.0, 31.0] ms.

In addition, a sensitivity analysis was conducted. We tested the performance of the DQN algorithm under different network loads (ranging from 20% to 100% of the maximum load) and various network topologies. The results showed that the DQN algorithm maintained relatively stable performance in terms of throughput, latency, and packet loss rate across different network loads and topologies. For instance, when the network load increased from 50% to 80% in the mesh topology, the throughput of the DQN algorithm decreased by only 5%, while the latency increased by 10%. This indicates the robustness of the DQN algorithm in different network environments.

These data will be used for subsequent result analysis and comparison to evaluate the performance differences of different algorithms under different traffic patterns.

This section comprehensively evaluates the performance of the traffic scheduling algorithm based on deep reinforcement learning (DQN) in three different network scenarios, including uniform load, dynamic load, burst traffic, and comprehensive scenarios, and compares it with traditional static routing algorithms and load balancing algorithms.

In addition to comparing with traditional static routing and load balancing algorithms, we also compared the DQN-based traffic scheduling algorithm with more advanced machine learning-based traffic optimization methods. For the long short-term memory network (LSTM) used for traffic prediction, we built an LSTM-based traffic scheduling model that uses historical traffic data to predict future traffic and make routing decisions based on it. In the same experimental scenario, when dealing with complex dynamic traffic, the LSTM model can predict traffic changes to a certain extent, but in terms of throughput, compared with the DQN-based algorithm, in the mesh topology and dynamic load scenario, the throughput of the LSTM model is 13.5 Gbps, which is lower than the 15.7 Gbps of the DQN algorithm. In terms of latency, the LSTM model has a latency of 35.6 ms in the burst traffic scenario, which is higher than the 25.7 ms of the DQN algorithm.

At the same time, we introduced two reinforcement learning variants, the proximal policy optimization (PPO) and the asynchronous advantage actor-critic algorithm (A3C), for comparison. The PPO algorithm improves learning efficiency by optimizing the policy network, while the A3C algorithm speeds up training through an asynchronous update mechanism. Experimental results show that under large-scale network topologies, the standard deviation of the load balancing effect of the PPO algorithm is 0.12, which is higher than the 0.08 of the DQN algorithm; the packet loss rate of the A3C algorithm under high load reaches 0.20%, while that of the DQN algorithm is 0.12%. These comparison results further highlight the advantages of the DQN-based traffic scheduling algorithm in multiple performance indicators.

4.5 Performance optimization and improvement directions

From the above experimental results, it can be seen that the DQN-based traffic scheduling algorithm is significantly superior to traditional static routing and load balancing methods in terms of throughput, delay, packet loss rate and load balancing. However, although the DQN algorithm has shown strong adaptability and robustness in most scenarios, there are still some bottlenecks, especially in burst traffic scenarios, the algorithm's delay and packet loss rate sometimes fluctuate. To address these issues, the following optimization directions can be considered:

(1) Transfer learning: Transfer learning enables the DQN algorithm to adapt to new environments more quickly, especially in bursty traffic situations, shortening the learning and adjustment time.
(2) Multi-path selection: Add multi-path traffic scheduling strategy to further reduce latency and packet loss rate by selecting more network paths for traffic distribution.
(3) Hybrid algorithms: Combine DQN with traditional algorithms (such as dynamic routing or congestion control algorithms) to form a hybrid traffic scheduling method to improve stability under extreme traffic conditions.

The SDN controller serves as the core of the

system. We choose the OpenDaylight controller. During implementation, the OpenDaylight software needs to be installed on the server and configured accordingly to enable it to communicate with the switches in the network. By configuring the interface parameters of the controller, ensure that it can accurately receive network state information from the switches, such as bandwidth utilization and link latency.

The intelligent traffic management module is written in Python and uses machine - learning and deep - learning related libraries (such as TensorFlow or PyTorch) to implement traffic prediction and scheduling algorithms. This module is deployed on the same server as the SDN controller and interacts with the controller through an internal interface. For example, after obtaining network state data from the controller, use the trained model to predict traffic and return the generated scheduling strategy to the controller.

The switches in the network topology adopt hardware switches that support the OpenFlow protocol. During deployment, the switches need to be initialized and configured, and the parameters for their communication with the SDN controller, such as the IP address and port number of the controller, need to be set. Ensure that the switches can forward data according to the flow table rules sent by the controller.

The traffic prediction and scheduling module is closely integrated with the intelligent traffic management module. When implementing traffic prediction, historical network traffic data and real - time network state data are used for model training. For example, a Long Short - Term Memory (LSTM) model is used to learn from historical traffic data and predict future traffic trends. The scheduling module then generates specific traffic scheduling strategies, such as choosing the optimal routing path and allocating bandwidth, based on the prediction results and the current network state.

The data forwarding module is mainly implemented by the switches. The switches forward network data according to the flow table rules issued by the SDN controller. During the integration process, ensure the accurate issuance and timely update of the flow table rules to adapt to changes in the network state. Through these specific implementation and integration methods, the SDN - based intelligent traffic management system can operate effectively to achieve intelligent management and optimization of network traffic.

4.6 Discussion

In this section, we directly compare the results of the DQN-based traffic scheduling algorithm with the current state-of-the-art (SOTA) technology.

Comparison of numerical results:

Latency: The experimental results show that in dynamic load scenarios, the latency of the DQN-based algorithm is 22.1 ms, while some state-of-the-art algorithms that rely on static configuration may have a latency of up to 40.5 ms. In burst traffic scenarios, the latency of the DQN algorithm is 25.7 ms, which is significantly lower than many traditional algorithms and some existing state-of-the-art methods. This shows that the DQN algorithm can better adapt to traffic fluctuations and reduce the transmission time of data packets.

Throughput: In a mesh topology, the DQN-based algorithm has a throughput of 15.7 Gbps, which is much higher than the 11.6 Gbps of static routing and 13.0 Gbps of load balancing, and is also better than some state-of-the-art algorithms that do not fully utilize real-time network status information for scheduling. This shows that the DQN algorithm can effectively improve the data transmission rate in complex network topologies.

Packet loss rate: Under high load conditions, the packet loss rate of the DQN algorithm is 0.12%, while some traditional and state-of-the-art algorithms may have a packet loss rate of up to 0.15% or even higher. This demonstrates the ability of the DQN algorithm to optimize traffic scheduling and reduce packet loss under challenging network conditions.

Reasons for superiority:

The DQN-based algorithm outperforms many existing methods, mainly because it can continuously learn from the real-time network environment. The use of deep neural networks in DQN enables it to handle high-dimensional and complex network state spaces. For example, in the state space, it comprehensively considers parameters such as bandwidth utilization, link delay, packet loss rate, and queue length. By interacting with the environment and adjusting the traffic scheduling strategy according to the reward function, the DQN algorithm is able to make smarter decisions compared to traditional static rule-based methods. In contrast, traditional methods usually rely on fixed rules or preset parameters and cannot adapt to the dynamic changes of the network environment in a timely manner.

Analysis of potential weaknesses and improvement directions:

Computational complexity: Although the DQN algorithm shows good performance, its computational complexity is relatively high. Training the deep neural network in DQN requires a lot of computing resources, which may limit its application in some resource-constrained network devices. Future research can focus on developing more efficient neural network architectures or training algorithms to reduce the computational burden.

Reward function design: The current reward function takes into account throughput, latency, and packet loss rate. However, the selection of related weight coefficients is relatively empirical. The optimal values of these weight coefficients may be different in different network scenarios. Therefore, more research is needed to develop a method to adaptively adjust these

weight coefficients according to the actual network situation.

Scalability in very large-scale networks: In very large-scale networks with a large number of nodes and complex topologies, the current distributed learning and collaborative optimization frameworks may face challenges in information exchange and global consistency maintenance. Further research is needed to improve the scalability of the algorithm in such scenarios.

4.7 Computational complexity analysis

The proposed deep reinforcement learning - based traffic scheduling method, specifically the DQN algorithm, has certain computational complexity. The DQN algorithm uses a deep neural network to approximate the Q - value function. The forward and backward propagation processes in the neural network contribute to the computational cost.

In terms of the number of parameters in the neural network, if we assume a simple feed - forward neural network structure with input neurons, hidden neurons, and output neurons, the number of parameters between the input and hidden layers is (including biases), and between the hidden and output layers is . For our traffic scheduling model, considering the state space dimensions (such as bandwidth utilization, link delay, etc., which might contribute to a relatively large number of input neurons), the number of parameters can be substantial.

During the training process, for each training episode, the agent interacts with the environment, and the Q - value function is updated. The time complexity of each Q - value update is related to the complexity of the neural network operations. With a learning rate of 0.001 and a batch size of 64, the computational cost per update is non - trivial.

In real - time applications, although the training time of approximately 24 hours on our experimental server (Intel Xeon Platinum 8280 processor, 512GB of RAM, and an NVIDIA Tesla V100 GPU) is a significant factor, once the model is trained, the inference time for making traffic scheduling decisions is relatively short. For example, in a real - time network with a moderate number of traffic flows, the DQN - based model can make a scheduling decision within a few milliseconds, which indicates its potential feasibility for real - time applications. However, in extremely large - scale real - time networks with high - frequency traffic changes, further optimizations might be required to reduce the computational overhead.

4.8 Scalability considerations

To evaluate the scalability of the proposed DQN - based traffic scheduling method, we conducted additional experiments on larger - scale networks. We increased the number of network nodes from the original 10 - 20 nodes in the previous experiments to 100 nodes in a more complex mesh - like topology.

As the number of nodes increased, the network traffic patterns became more complex, with a greater number of source - destination pairs and higher traffic volumes. The results showed that the throughput of the DQN - based algorithm decreased by 15% when the number of nodes increased from 20 to 100. The latency increased from an average of 20 ms to 30 ms. In terms of the load balancing effect, the standard deviation of the load distribution among nodes increased from 0.05 to 0.10.

When considering dynamic user behavior, we simulated scenarios where users' traffic demands changed rapidly. For example, in a scenario where 30% of users suddenly increased their traffic requests by 50%, the DQN - based algorithm was able to adjust the traffic scheduling, but the packet loss rate increased from 0.1% to 0.2%. These results indicate that while the DQN - based method can still function in larger - scale networks and dynamic user behavior scenarios, there is a certain degree of performance degradation, and further optimizations are needed to improve its scalability.

4.9 Practical deployment considerations

In practical implementation, the DQN - based traffic scheduling method faces several challenges.

Regarding real - time adaptability, in real - world networks, traffic patterns can change rapidly. The DQN algorithm needs to be able to update its traffic scheduling decisions in a timely manner. Although the current algorithm can make decisions within a few milliseconds after training, the time interval between traffic pattern changes might be even shorter in some high - speed networks. To address this, we might need to optimize the model's update mechanism to reduce the time required for re - evaluating the network state and making new decisions. The software - defined network (SDN) controller also has limitations. The SDN controller in our experiments was able to manage the network state and issue control commands. However, in large - scale real - world deployments, the controller might face performance bottlenecks when handling a large number of network devices and high - volume traffic data. For example, if there are thousands of network switches, the controller might experience delays in collecting network status information and sending control instructions. In terms of performance under real - world network traffic patterns, real - world traffic often has more complex characteristics than the simulated traffic in our experiments. There might be long - tailed distributions of traffic volumes, and sudden bursts of traffic from specific applications. The DQN - based algorithm needs to be further tested and optimized to ensure stable performance in such real - world scenarios.

5 Conclusion

This paper proposes an intelligent traffic scheduling algorithm based on deep reinforcement learning (DQN), and conducts experimental verification in different network scenarios to evaluate its performance and advantages. The experimental results show that the traffic scheduling algorithm based on DQN has significant improvements in multiple key performance indicators compared with traditional static routing and load balancing algorithms. Specifically, the DQN algorithm shows strong advantages in throughput, delay, packet loss rate and load balancing effect. Especially in dynamic load and burst traffic scenarios, DQN can quickly adapt to changes and adjust traffic paths, thus avoiding the bottlenecks in traditional methods. In different scenarios such as uniform load, dynamic load and burst traffic, the scheduling algorithm based on DQN can always provide low delay and packet loss rate, high throughput, and can effectively balance the traffic distribution in the network. Especially in burst traffic scenarios, the traditional static routing algorithm often leads to network overload due to its lack of flexibility, resulting in large packet loss rate and high delay. Although the load balancing algorithm can alleviate this problem to a certain extent, it still cannot provide the same performance as the DQN algorithm under high load. In addition to its advantages in basic performance, the DQN algorithm also demonstrates its strong adaptability and robustness, especially in the face of changes in network topology and load fluctuations, it can continuously adjust the traffic path to ensure the stability and efficient operation of the network. This feature makes the DQN algorithm have great application potential in the field of intelligent traffic management, especially for high-speed, high-load and frequently changing network environments.

In addition to its advantages in basic performance, the DQN algorithm also demonstrates its strong adaptability and robustness, especially in the face of changes in network topology and load fluctuations, it can continuously adjust the traffic path to ensure the stability and efficient operation of the network. This feature makes the DQN algorithm have great application potential in the field of intelligent traffic management, especially for high - speed, high - load and frequently changing network environments. However, as mentioned in the discussion, transfer learning is a potential optimization method that has not been experimentally evaluated in this study. In future work, we plan to conduct experiments on transfer learning. For example, we will first train the DQN model in a simulated network environment with a certain set of traffic patterns and network topologies. Then, we will attempt to transfer the learned knowledge to a new, real - world - like network environment with different but related traffic characteristics. By comparing the performance of the DQN model with and without transfer learning in the new environment, we can evaluate the effectiveness of transfer learning in improving the algorithm's adaptability and reducing the training time in new scenarios.

References

[1] Bao K, Matyjas JD, Hu F, Kumar S. Intelligent software-defined mesh networks with link-failure adaptive traffic balancing. IEEE Transactions on Cognitive Communications and Networking. 2018; 4(2):266-76. https://doi.org/10.1109/tccn.2018.2790974

[2] Malboubi M, Peng SM, Sharma P, Chuah CN. A learning-based measurement framework for traffic matrix inference in software defined networks. Computers & Electrical Engineering. 2018; 66:369-87. https://doi.org/10.1016/j.compeleceng.2017.11.020

[3] Huang R, Guan WF, Zhai GT, He JH, Chu XL. Deep graph reinforcement learning based intelligent traffic routing control for software-defined wireless sensor networks. Applied Sciences-Basel. 2022; 12(4):21. https://doi.org/10.3390/ app12041951

[4] Liu L, Zhou JT, Xing HF, Guo XY. Flow splitting scheme over link-disjoint multiple paths in software-defined networking. Concurrency and Computation-Practice & Experience. 2022; 34(10):18. https://doi.org/10.1002/cpe. 6793

[5] Tam P, Math S, Kim S. Intelligent massive traffic handling scheme in 5G bottleneck backhaul networks. KSII Transactions on Internet and Information Systems. 2021; 15(3):874-90. https://doi.org/10.3837/tiis.2021.03.004

[6] Keshari SK, Kansal V, Kumar S. An intelligent way for optimal controller placements in software-defined-IoT networks for smart cities. Computers & Industrial Engineering. 2021; 162:9. https://doi.org/10.1016/j.cie.2021.107667

[7] Zhao L, Bi ZG, Lin MW, Hawbani A, Shi JL, Guan YC. An intelligent fuzzy-based routing scheme for software-defined vehicular networks. Computer Networks. 2021; 187:13. https://doi.org/10.1016/j.comnet.2021.107837

[8] Guo YY, Wang WP, Zhang H, Guo WZ, Wang ZL, Tian Y, et al. Traffic Engineering in hybrid software defined network via reinforcement learning. Journal of Network and Computer Applications. 2021; 189:12. https://doi.org/10.1016/j.jnca.2021.103116

[9] Casas-Velasco DM, Rendon OMC, da Fonseca NLS. DRSIR: A deep reinforcement learning approach for routing in software-defined networking. IEEE Transactions on Network and Service Management. 2022; 19(4):4807-20. https://doi.org/10.1109/ tnsm.2021.3132491

[10] Guo X, Xian HB, Feng T, Jiang YB, Zhang D, Fang

JL. An intelligent zero trust secure framework for software defined networking. PeerJ Computer Science. 2023; 9:37. https://doi.org/10.7717/peerj-cs.1674

[11] Pitchai MP, Ramachandran M, Al-Turjman F, Mostarda L. Intelligent framework for secure transportation systems using software-defined-internet of vehicles. CMC-Computers Materials & Continua. 2021; 68(3):3947-66. https://doi.org/10.32604 /cmc.2021.015568

[12] Smida K, Tounsi H, Frikha M. Intelligent and resizable control plane for software defined vehicular network: a deep reinforcement learning approach. Telecommunication Systems. 2022; 79(1):163-80. https://doi.org/10.1007/s11235-021-00838- 2

[13] Wu CQ, Zhang YL, Li N, Rezaeipanah A. An intelligent fuzzy-based routing algorithm for video conferencing service provisioning in software defined networking. Telecommunication Systems. 2024; 87(4):887-98. https://doi.org/10.1007/s11235-023 -01044-y

[14] Lei JR, Deng SH, Lu ZB, He YH, Gao XP. Energy-saving traffic scheduling in backbone networks with software-defined networks. Cluster Computing-the Journal of Networks Software Tools and Applications. 2021; 24(1):279- 92. https://doi.org/10.1007/s10586-020-03102-5

[15] Guo AP, Yuan CH. Network intelligent control and traffic optimization Based on SDN and artificial intelligence. Electronics. 2021; 10(6):18. https://doi.org/10.3390/electronics10060700

[16] Prasanth LL, Uma E. A computationally intelligent framework for traffic engineering and congestion management in software-defined network (SDN). EURASIP Journal on Wireless Communications and Networking. 2024; 2024(1):22. https://doi.org/10.1186/s13638-024- 02392-2

[17] Huo LW, Jiang DD, Lv ZH, Singh S. An intelligent optimization-based traffic information acquisition approach to software-defined networking. Computational Intelligence. 2020; 36(1):151-71. https://doi.org/10.1111/coin.12250

[18] Nam C, Math S, Tam P, Kim S. Intelligent resource allocations for software-defined mission-critical IoT services. CMC-Computers Materials & Continua. 2022; 73(2):4087-102. https://doi.org/10.32604/cmc.2022.030575

[19] Kumar A, Anand D, Jha S, Joshi GP, Cho W. Optimized load balancing technique for software defined network. CMC-Computers Materials & Continua. 2022; 72(1):1409-26. https://doi.org/10.32604/cmc.2022.024970

[20] Casas-Velasco DM, Rendon OMC, da Fonseca NLS. Intelligent routing based on reinforcement learning for software-defined networking. IEEE Transactions on Network and Service Management. 2021; 18(1):870-81. https://doi.org/10.1109/tnsm.2020.3036911

[21] Balakiruthiga B, Deepalakshmi P. (ITMP)-intelligent traffic management prototype using reinforcement learning approach for software defined data center (SDDC). Sustainable Computing-Informatics & Systems. 2021; 32:19. https://doi.org/10.1016/j.suscom.2021.100610

[22] Modi TM, Swain P. Intelligent routing using convolutional neural network in software-defined data center network. Journal of Supercomputing. 2022; 78(11):13373-92. https://doi.org/10.1007/s11227-022-04348-z