Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **10**, Number **3**, August **2025**

SDN Optimal Path Decision Algorithm Integrating Multiple Network Indicators

Yu-Xin Wang, Ying-Li Wang, Qing-Huan Xu, Hong-Bin Ma*

School of Electronic Engineering
Heilongjiang University, Harbin 150006, China
2221718@s.hlju.edu.cn, wangyingli@hlju.edu.cn, 2221744@s.hlju.edu.cn, mahongbin@hlju.edu.cn

Jia-Ni Wei

School of Mathematics
University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom
jxw1615@student.bham.ac.uk

*Corresponding author: Hong-Bin Ma

Received June 17, 2024, revised October 29, 2024, accepted February 20, 2025.

Abstract. *This paper studies the path decision problem in Software Defined Network (SDN). At present, the deep learning-based SDN path decision model still faces certain challenges in terms of dynamic network indicator integration, decision accuracy and deployment scalability. To address the above issues, we constructed an SDN-based path decision model that integrates multiple network indicators. First, we introduced the Gate Recurrent Unit (GRU) and integrated it with SDN, combining it with attention to improve the performance of the decision-making model. Second, we took comprehensive consideration of network indicators (delay, loss tolerance and bandwidth) and realized dynamic update and comprehensive consideration of these indicators through methods such as normalization and fusion. Then, we map the fused network indicators to the weight of each link, and use this to identify the best route for transmitting traffic within the network to ensure the Quality of Service (QoS). Ultimately, we perform experiments on three different topologies, and the outcomes demonstrate our model considerably enhances the three network indicators in comparison to the default shortest path. Compared with existing path decision algorithms, our approach comprehensively considers network indicators and improves network reliability.*
Keywords: Software Defined Network, Multiple Network Indicators, Path Decision, GRU, Attention

1. **Introduction.** 5G has changed network requirements due to the development of Internet intelligent terminal systems and the emergence of concepts such as cloud computing and the Internet of Things [1]. The combination of IoT devices with cloud computing considerably promotes resource sharing, facilitates users' access to information at any time, and provides users with corresponding cloud computing services on time [2]. Software Defined Networking (SDN) is a new architecture that supports new network requirements [3]. The advent of SDN has revolutionized the way we manage and control modern network infrastructure. Decoupling control plane from data plane, SDN provides greater programmability and scalability, making it key network technology. One challenge facing SDN is how to make intelligent path decisions, which relies on accurate analysis of network indicators and status. The principle of routing decision is to select an optimal path based on routing information and forward the data. Determining the best path for data

to be transmitted is a key challenge. Traditional routing algorithms usually rely on static metrics, which are incapable of coping with dynamic changes in network traffic and can easily lead to network congestion and bottleneck problems [4], thus affecting the QoS. This challenge motivates researchers to seek new methods to optimize routing decisions through dynamic updating of real-time network indicators.

This paper proposes a dynamic path decision method based on the GRU prediction model and attention mechanism to cope with the increasingly complex demands and frequent changes in network status in modern network environments. With the widespread application of the Internet of Things, cloud computing, and intelligent terminal systems, the complexity of network traffic has increased significantly, and traditional routing algorithms that rely on static indicators have difficulty coping with such changes. Therefore, this paper combines the time series prediction capability of GRU with the attention mechanism, comprehensively considering key network indicators such as delay, loss tolerance, and bandwidth, aiming to accurately analyze and predict the future state of the network and dynamically adjust the path decision strategy to ensure that data can be transmitted efficiently and stably.

1.1. **Realted work.** With the advancement of deep learning, it has shown great potential in solving network problems. Currently, more and more researchers are beginning to explore the combination of deep learning and SDN to achieve more optimized routing decisions and improve network performance. Many studies [5–10] have proposed decision-making models based on reinforcement learning. However, these methods fail to fully address the challenges of integrating dynamic network indicators and have low prediction accuracy in large-scale networks. In addition, it is difficult to deploy routing strategies for large-scale networks driven by reinforcement learning, their scalability in network topologies is also limited. To solve the scalability problem in different network topologies, some studies [11–15] introduced Graph Neural Network (GNN) into SDN path decision algorithms. GNN has good generalization ability for different topologies and can capture the relationship between different nodes and links. However, GNN have difficulty converging during training and face obstacles in practical deployment. These factors have led to the fact that GNN has not been commonly applied in the realm of path decision. Addressing the constraints of the aforementioned deep learning models in SDN, we integrate the Gate Recurrent Unit [16] in Recurrent Neural Network with SDN. This integration not only solves the problem that reinforcement learning is difficult to deploy in SDN, but also makes full use of the advantages of GRU as a variant of RNN, especially in solving the gradient disappearance and explosion problems that plague traditional RNN. In this way, deep learning models can be well integrated with SDN to improve network performance and decision effectiveness. Some studies have also integrated the GRU model with SDN, using the GRU's predictive capabilities to predict future network indicators based on real-time network indicators to make path decisions. However, these studies do not take into account that network indicators at different moments have different importance for current state prediction. To improve this, we optimized the existing model and introduced an attention mechanism to assess the influence of network indicators across various time points on the current state prediction. In this way, we are able to more accurately predict the network state, thus enhancing the performance of overall network.

Besides, and most importantly, current methods lack comprehensive consideration of delay, loss tolerance, and bandwidth, which are key network indicators in path decision making. In order to achieve more effective and accurate path decisions, these key indicators need to be systematically integrated and analyzed. Some works [17–20] attempt to incorporate specific indicators into path decision algorithms. For example, work [18]

mainly considers link utilization, work [20] focuses on reducing delay, this paper is also based on these two works, and many works focus on reducing loss tolerance. However, these methods lack comprehensive consideration of all network indicators, and the optimization of a single specific indicator has limited effect on improving the overall network performance. To overcome these shortcomings, we systematically integrate and analyze multiple network indicators to achieve more comprehensive and accurate path decisions.

1.2. **Motivation and contribution.** This paper primary works are as follows: (a) Integrate the prediction capability of GRU into SDN, introduce attention mechanism to take into account the different importance of network indicators at different times to the current state prediction, improve the performance of the overall decision model, and enable it to more accurately predict the network state and make corresponding path decisions. (b) We comprehensively consider the key network indicators of delay, loss tolerance and bandwidth in path decision-making, normalize and fuse these indicators, map the fused network indicators to the weight of each link and dynamically update them, and use these weights to decide optimal path for the traffic to be transmitted in the network.

2. Methodology.

2.1. **Problem Statement.** SDN is abstracted as G=(V,E), among them V denoting nodes and E denoting links [21]. GRU is used to predict future indicators in the network, including delay, loss tolerance, and bandwidth. The path decision for the traffic to be transmitted in the network is made based on the predicted indicators. Real-time delay, loss tolerance, and bandwidth, represented by D, Lt, and B individually. The normalized delay, loss tolerance and bandwidth are denoted as $norm\_D$, $norm\_Lt$, $norm\_B$. In addition, $pred\_D$, $pred\_Lt$, and $pred\_B$ represent future indicators predicted by the decision model, which are used to make path decisions for traffic to be transmitted in the network.

The GRU model can be represented as: If $x(t)$ represents the input sequence of network indicators at time t, and $y(t)$ represents the output sequence of network indicators at time t.

$$y(t) = \mathrm{GRU}(x(t), h(t-1)) \tag{1}$$

where $h(t-1)$ is the hidden state of the GRU model at time $t-1$.

Expressed as the decision problem is: W denote the weight of link $e \in E$, P represent the set of paths in G, R denote the routing forwarding path set by the controller, and F represent the flow set in SDN.Expressed as the decision problem is: W denote the weight of link $e \in E$, P represent the set of paths in G, R denote the routing forwarding path set by the controller, and F represent the flow set in SDN.

(1) Obtain the real-time delay D, loss tolerance Lt, and bandwidth B of each link from the host application.

(2) Normalize the indicators in (1) to get $norm\_D$, $norm\_Lt$, and $norm\_B$.

(3) To predict future delay, loss tolerance, and bandwidth, utilize the normalized indicators from (2).

$$pred\_D, pred\_Lt, pred\_B = \mathrm{GRU}(norm\_D, norm\_Lt, norm\_B) \tag{2}$$

(4) The link weight $W(e)$ is determined using the prediction indicator in (3). $W(e)$ is mapped to the arithmetic means of the predictors.

$$W(e) = \frac{(pred\_D + pred\_Lt + pred\_B)}{3} \tag{3}$$

(5) The path with the minimum weight is obtained using the weighted shortest path algorithm [22]. In G, the weights $W(e)$ of the edges from source node s to destination node d determine a weighted shortest path, the path length is calculated as the sum of the link weights. Denote the set of all paths from s to d in G as $P(s,d)$, and the shortest path from s to d as $SP(s,d)$. The weighted shortest path from s to d can be expressed as:

$$\min SP(s,d) \tag{4}$$

$$SP(s,d) = W(e_1) + W(e_2) + \cdots + W(e_k) \tag{5}$$

(6) The path R of the installation decision is made in the data plane. The overall objective function is to reduce the overall delay (O_D), the overall loss tolerance (O_{Lt}) and increase the overall bandwidth (O_B).

$$\min \sum f \in F(O_D, O_{Lt}, \frac{1}{O_B}) \tag{6}$$

2.2. **Path Decision Model.** The goal of this paper is to obtain real-time network indicators, normalize them and use them to predict future indicators to make optimal path decisions for traffic to be transmitted. Figure 1 depicts overall framework of the decision model. First, the controller receives topology through topology information module. Second, real-time delay, loss tolerance, and bandwidth are got through port statistics module. Normalization of these indicators is performed to maintain uniformity and comparability across various indicators and time periods. The weight of each link in SDN is determined using the predicted future indicators. Following the determination of the link weights, the optimal path from source node to destination node is identified using a weighted shortest path algorithm.

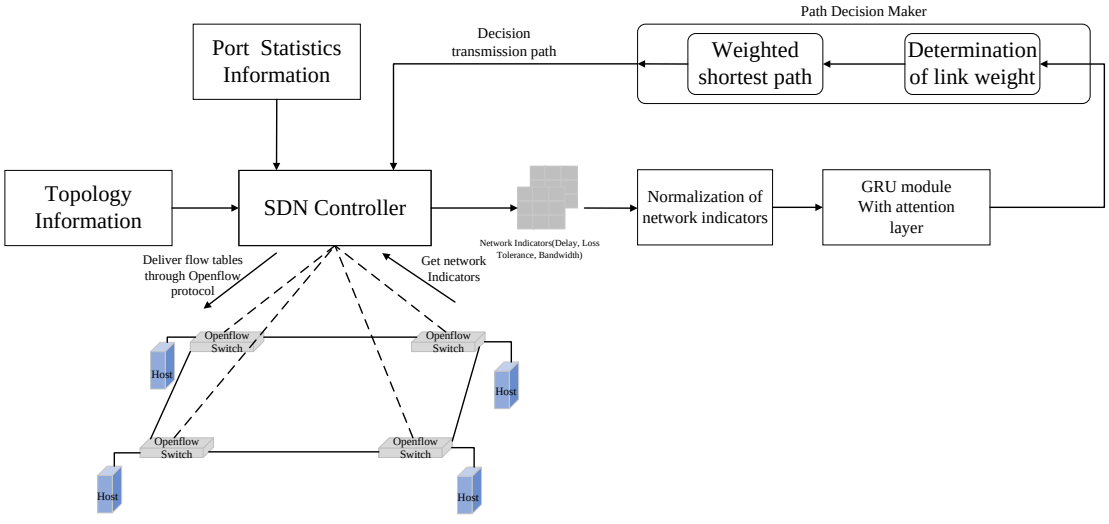

FIGURE 1. Overall framework of decision model

Figure 2 shows the workflow. In the form of a matrix, network indicators are collected and represented, where $V(t)$ represents the indicators collected from the host at time t, the rows of the matrix represent delay, loss tolerance, and bandwidth, and links are represented by the columns. $V(t-4)$, $\cdots$, $V(t)$ represent measurement values on continuous

time series. The measurement values have different types and ranges and need to be normalized. The normalized measurement value is represented by $NV(t)$. Using the input sequence and hidden state, the trained GRU model predicts the indicator at time $t+1$ according to the previous 5 time series indicators [23]. From an empirical perspective, indicators at different times have different weights on the current state prediction. Therefore, the GRU model needs to assess the influence of network indicators across various time points, attention mechanism is incorporated in hidden layer of the GRU to calculate the attention distribution value of the GRU hidden state at each moment, and then weighted sum each hidden state with the corresponding attention distribution value. The output of the previous attention layer serves as the input to the output layer. The weight of the link is determined by predicting the indicators through the GRU model with an attention layer. To decide the optimal routing path, apply the weighted shortest path algorithm and determine the weights. The controller will install the selected path in the flow table and send it down. Based on the predicted indicators, periodic updates will be made to the link weights, and optimal path from source node to destination node may be different at different times, depending on the predicted indicators optimized at different times.

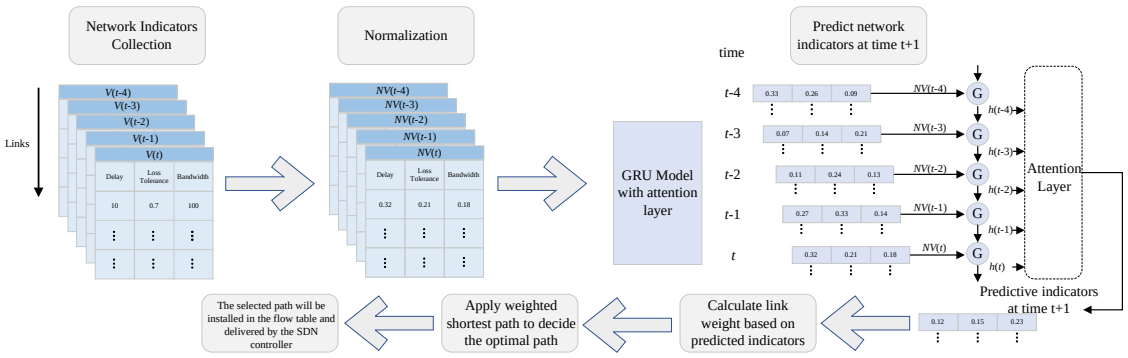

FIGURE 2. Overall process of decision model

2.2.1. *Network Indicator Collection Design.* (1) Measurement topology: In SDN, we use the OpenFlow Discovery Protocol (OFDP) to measure network topology. The measurement process uses Packet-Out and Packet-In messages. The measurement principle is shown in Figure 3. The specific process is: a) The controller constructs a Packet-Out message to send an LLDP data packet to S1. b) The controller sends a flow table to S1, with the rule that send the LLDP data packets received from the controller out of the corresponding port. c) The controller sends a flow table to S2, with the rule that LLDP packets received from non-controllers are sent to the controller. d) The controller obtains information about S1 forwarding the data packet to S2 through the Packet-In message, including port information, switch ID, etc. The data packet path and network topology are determined based on the port information and switch ID, and the topology of the entire network is then inferred.

(2) Measuring link delay: Since OpenFlow switches do not include timestamps in normal data packets, delay cannot be measured passively like traditional networks. Therefore, we need to generate and send probe packets between switches to measure the delay. The delay measurement process is shown in Figure 4. The specific process is: a) Assume that

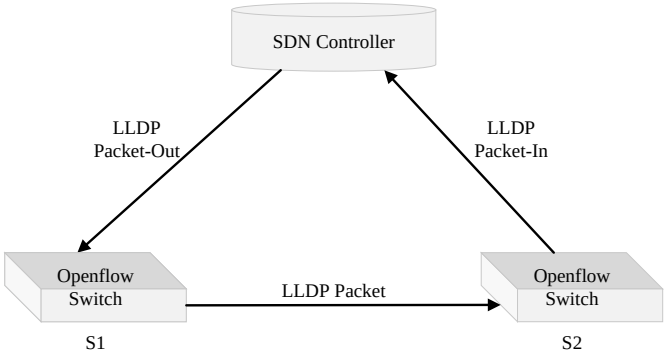

FIGURE 3. Principles of topology measurement

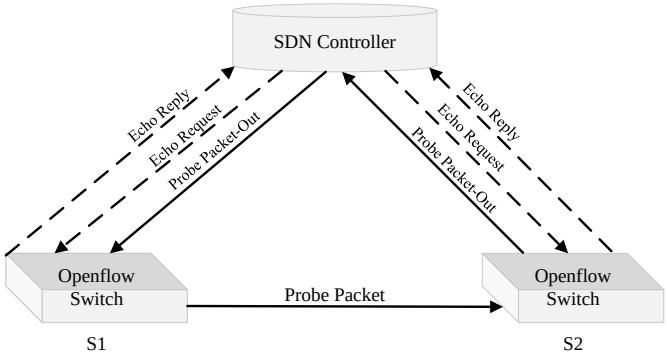

FIGURE 4. Principles of link delay measurement

the transmission time of the probe packet is T_{tt}. In order to obtain T_{tt}, the controller generates a probe data packet containing the S1 destination forwarding port and the sending timestamp, and records the transmission path and sending time. These data packets will be encapsulated into Packet-In messages and sent to S1. S1 will forward the data packet to the specified port, which will then be received by S2. When S2 receives a data packet, it lacks a matching flow table entry, triggers a Packet-In message, and encapsulates the data packet and returns it to controller. The controller uses the timestamp of the data packet and the time when the data packet is received to calculate the total time T_{tt} for the data packet to be transmitted on the path of controller→S1→S2→controller. b) Assume that the round trip time between the controller and the switch is T_{rt}. The controller generates Echo Request message and sends it to S1 and S2, waits for the Echo Reply message to be received, and records the timestamp of the received message. $T_{rt\mathrm{S1}}$ and $T_{rt\mathrm{S2}}$ are obtained. c) According to a) and b), the time delay T_d between S1 and S2 can be calculated:

$$T_d = T_{tt} - \frac{T_{rt\mathrm{S1}} + T_{rt\mathrm{S2}}}{2} \tag{7}$$

(3) Measuring link loss tolerance: In SDN, Port-Stats messages are mainly used to measure link loss tolerance and bandwidth. There are two types of Port-Stats messages: one is the Port-Stats-Request message, which is used by the controller to request the switch port statistics information, and the other is the Port-Stats-Reply message, which is used by the switch to respond to the controller. The Port-Stats-Reply message includes the number of packets received/ transmitted ($rx\_packets$/ $tx\_packets$), the number of bytes received/ transmitted (rx_bytes/ tx_bytes), the number of dropped packets ($rx\_dropped$/

tx_dropped), the number of collisions (*collisions*), the port lifetime (*duration_sec* and *duration_nsec*). In the topology shown in Figure5 , the loss tolerance of the link from S1 to S2 is measured. Use the Port-Stats-Request message to obtain the number of packets transmitted by port 1 of S1 ($tx_packets$S1) and the number of packets received by port 2 of S2 ($rx_packets$S2). In this paper, we measure the loss tolerance every 5s and use the following formula to calculate the loss tolerance within two query time periods.

$$LT(t-1,t) = 1 - \frac{rx\_packets_{\mathrm{S2}(t)} - rx\_packets_{\mathrm{S2}(t-1)}}{tx\_packets_{\mathrm{S1}(t)} - tx\_packets_{\mathrm{S1}(t-1)}} \tag{8}$$

among them, $LT(t-1,\ t)$ represents the loss tolerance from the $t-1$-th query to the t-th query time period, $rx\_packets_{\mathrm{S2}(t)}$ represents the number of packets received by S2 in the t-th query, the rest of the parameters are the same.

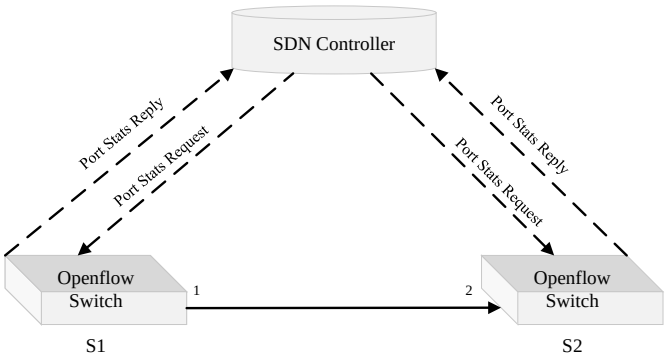

FIGURE 5. Principles of link delay measurement

(4) Measuring link bandwidth: Bandwidth measurement requires measuring link throughput and port data transmission rate. When measuring the port data transmission rate, the controller sends a Port-Stats-Request message to the specified switch and obtains the received data stream bit size (*byte_count*) and port lifetime (*duration_sec* and *duration_nsec*) from the Port-Stats-Reply message replied by the switch. The actual transmission rate of the data stream is as follows:

$$Speed = \frac{byte\_count}{duration\_sec + duration\_nsec \times 10^{-9}} \times 8 \tag{9}$$

When measuring link throughput, Periodically, Port-Stats-Request messages are sent by the controller to the designated switch and obtains the number of received/ transmitted bytes (rx_bytes/ tx_bytes) and port lifetime (*duration_sec* and *duration_nsec*) from the Port-Stats-Reply message replied by the switch. The throughput from the $t-1$-th measurement to the t-th measurement is:

$$duration = duration\_sec + duration\_nsec \times 10^{-9} \tag{10}$$

among them, *duration* represents the port lifetime, *duration_sec* represents the port lifetime in seconds, and *duration_nsec* represents the port lifetime in milliseconds.

$$Throughtput(t-1,t) = \frac{[(tx\_bytes_t + rx\_bytes_t) - (tx\_bytes_{t-1} + rx\_bytes_{t-1})]}{duration_t - duration_{t-1}} \times 8 \tag{11}$$

among them, $Throughput(t-1,\ t)$ is the throughput from the $t-1$-th measurement to the t-th measurement.

Upon obtaining the port data transmission rate and port throughput, one can calculate the available bandwidth $free\_bandwidth$ of the current link.

$$free\_bandwidth = Throughtput - Speed \tag{12}$$

2.2.2. *Normalization of Network Indicators.* As shown in Figure 6, the indicators obtained from the host are represented in the form of a matrix. The matrix columns represent specific network indicators, in this case delay, loss tolerance, and bandwidth. The rows represent specific links. Using vector normalization, normalize the matrix parameters. The dynamic nature of the parameters is indicated by entropy. Deviation and entropy are complementary. The larger the deviation, the greater the impact of network indicators on network performance. In this paper, the weight of each indicator is the contribution of the deviation of each indicator.

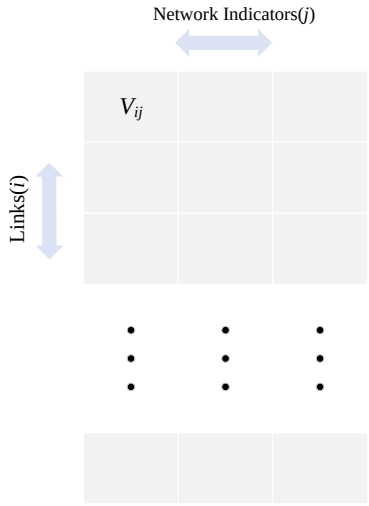

FIGURE 6. Representation of network indicators

The parameter vector is normalized as follows:

$$NV_{ij} = \frac{V_{ij}}{\sqrt{\sum_{i=1}^{I}(V_{ij})^2}} \quad 0 \le NV_{ij} \le 1, \quad 1 \le i \le I, \quad J = 1, 2, 3 \tag{13}$$

among them, the number of links is represented by I, network indicators of the i-th link are represented by V_{ij}, namely, delay, loss tolerance, and bandwidth. The nature of the change of each parameter is given by entropy,

$$E_j = -\frac{1}{\ln(I)} \times \left[\sum_{i=1}^{I}(NV_{ij} \times \ln(NV_{ij}))\right] \quad 0 \le E_j \le 1 \tag{14}$$

NV_{ij} represents the normalization of network indicators. Deviation and entropy are complementary. Deviation can be expressed as:

$$D_j = 1 - E_j \quad 0 \le D_j \le 1, \quad j = 1, 2, 3 \tag{15}$$

among them, the entropy of the j-th network indicator is represented by E_j. The calculation of indicator weights is as follows:

$$W_j = \frac{d_j}{\sqrt{\sum_{j=1}^{J} d_j}} \quad 0 \le W_j \le 1, \quad J = 3 \tag{16}$$

among them, The weight of the j-th indicator is represented by W_j, while the deviation of the j-th indicator is represented by D_j. At present consider the weight calculated for each parameter,

$$NV_{W_{ij}} = W_j \times NV_{ij} \quad 1 \le i \le I, \quad j = 1, 2, 3 \tag{17}$$

$NV_{W_{ij}}$ represents the weighted normalized index of link i.

2.2.3. *GRU Module and Attention Layer.* The reliability of decision models depends largely on the accuracy of network indicator predictions. Nevertheless, many shortcomings are often encountered by conventional prediction techniques in this regard. Additionally, RNN-based methods have exhibited exceptional abilities in capturing dependencies across different time intervals. This paper adopts a variant of RNN, the GRU, to solve the prediction problem. Figure 7 illustrates the internal structure of GRU, and the GRU model in this paper is shown in Figure 8.

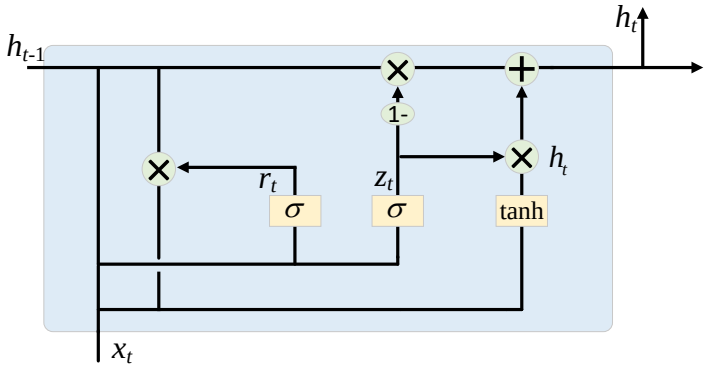

FIGURE 7. The internal structure of GRU

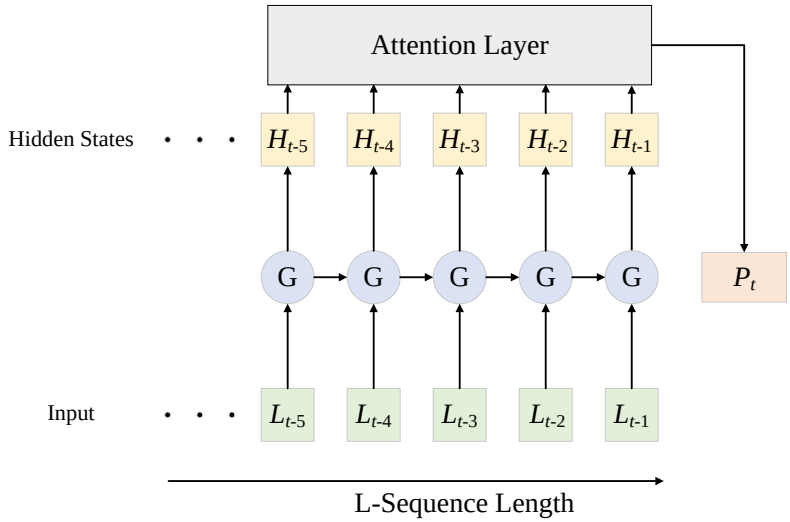

FIGURE 8. The GRU model of this paper

For the GRU module, given an input sequence x=$NV_{W_{ij}}$, among them, the total number of links is represented by I, the initial hidden state is represented as h_0, and h_1, h_2, $\cdots$, h_n are calculated by GRU as follows:

$$z_t = \sigma(x_t W_{xz} + h_{t-1} W_{hz} + b_z) \tag{18}$$

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r) \tag{19}$$

$$\widetilde{h_t} = \tanh(x_t W_{xh} + (r_t \odot h_{t-1} W_{hh} + b_h)) \tag{20}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{h_t} \tag{21}$$

learnable parameters include W_{xz}, W_{hz}, b_z, W_{xr}, W_{hr}, b_r, W_{xh}, W_{hh} and b_h, the tanh activation function is represented by tanh, while the Sigmoid activation function is represented by σ.

z_t represents the update gate, which regulates the retention of information from the previous hidden state h_{t-1} and the integration of the candidate state $\widetilde{h_t}$. r_t represents reset gate, which adjusts the degree to which the previous hidden state is neglected. The generation of $\widetilde{h_t}$ involves combining xt and previous hidden state h_{t-1}, which is then adjusted by r_t. The final hidden state ht is formed by weighting h_{t-1} and $\widetilde{h_t}$, and is controlled by z_t. This enables GRU to selectively retain or forget information from past hidden states and current inputs, thereby effectively capturing relationships in continuous data and completing the network indicator prediction task in this paper.

The indicator sequence generated by Section 2.2.1 serves as the input for GRU to forecast future indicators. L_i represents the indicator generated at time i in the time series. The predicted value P_t is used to map the link weight.

From experience, network indicators at different times have different impacts on the prediction at the current time. The GRU model needs to assess the influence of network indicators across various time points, an attention mechanism is introduced [24]. Calculate the attention weight of the GRU hidden state at each moment, and then sum the weighted sum of each hidden state and the corresponding attention weight. Introduce the attention mechanism in the hidden layer and calculate the attention distribution value a_1, a_2, $\cdots$, a_i of each input. Further extract the time features and highlight the important time.From experience, network indicators at different times have different impacts on the prediction at the current time. The GRU model needs to assess the influence of network indicators across various time points, an attention mechanism is introduced [24]. Calculate the attention weight of the GRU hidden state at each moment, and then sum the weighted sum of each hidden state and the corresponding attention weight. Introduce the attention mechanism in the hidden layer and calculate the attention distribution value a_1, a_2, $\cdots$, a_i of each input. Further extract the time features and highlight the important time.

$$e_i = W_{e1} \tanh(W_{e2} h_i + b_e) \tag{22}$$

$$a_i = \frac{\exp(e_i)}{\sum_{i=1}^{I} e_i} \tag{23}$$

$$v = \sum a_i h_i \tag{24}$$

among them, e_i is the attention value determined by the hidden layer state vector h_i at time i, W_{e1}, W_{e2}, b_e are learnable parameters. a_i is the weight calculated by the attention layer under each hidden state vector h_i. Then the attention weight values at each moment are weighted and summed, and the output of attention layer calculates the final feature vector containing the time information of the attention mechanism.

The input of output layer is the output of previous Attention layer. With an initialization of 1 hidden layer and 1024 hidden parameters, unidirectional GRU module ultimately generates 3 output values. When hidden sequences are combined, the output size is (N, L, D^*H), with N representing batch size, L denoting the sequence length, D set to 1 in unidirectional GRU, and H being the number of output parameters, which is 3. It is subsequently converted to a tensor of size(N, L^*D^*H). The transformed tensor is fed

into a fully connected network, which converts it to a size of (N, the number of network indices) and generates the prediction result.

2.2.4. *Path Decider.* Through Section 2.2.2, we normalize network indicators of different dimensions and ranges, and assign different weights to each indicator to obtain the normalized network indicators of each link. Now, we fuse the normalized network indicators of each link and map them into corresponding link weights. The path with the minimum weight is then identified using a weighted shortest path algorithm. A timeout period is designated for each path. Once this period expires, the process is repeated to refresh the weight and send different paths according to the traffic to be transmitted at the data layer.

Algorithm 1 Decision-making model

Input: Indicators $V(t)$ collected from the host at time t.
Output: Optimal path.
1: $NV(t) \leftarrow V(t)$ //Standardize and normalize $V(t)$ to obtain $NV(t)$.
2: The GRU model uses 5 consecutive time series $NV(t-4)$, $NV(t-3)$, $NV(t-2)$, $NV(t-1)$ to predict $NV(t+1)$.
3: Fusion of indicators of each link in $NV(t+1)$.
4: Map the fused network indicators to the weights of each link.
5: Utilize the weights of each link to apply the weighted shortest path algorithm.
6: **Return:** Optimal path.
7: This process is performed every 5 seconds. That is, the weight of each link is updated every 5 seconds.

The network indicators are integrated and mapped into link weights as follows:

$$W_t(i) = \sum_{j=1}^{3} P_{ij} \Big/ 3 \quad 1 \le i \le I \tag{25}$$

among them,$W_t(i)$represents the weight of the i-th link, P_{i1}, P_{i2} and P_{i3} represent the predicted link delay, loss tolerance, and bandwidth respectively.

The weighted shortest path algorithm is derived from the Dijkstra Algorithm. Its core is to seek the shortest path that minimizes the total weight between source node and destination node. The weights along the links are summed to determine the total path weight. The overall decision model is shown in Algorithm 1.

3. Experiment.

3.1. **Experiment environment.** The environment including a collection of virtual hosts, switches, and links, facilitated by the Mininet simulation platform. During the test, three different topologies were used to analyze the algorithm performance, namely NYC (New York City Center Network) [25], Ring, and Abilene, as shown in Figures 9, 10, and 11. The Ryu controller is used to manage the network topology, implement communication with switches, and issue flow tables.

There are two categories of traffic in the network: elephant flow and mouse flow. Therefore, the iperf tool is used to simulate and generate TCP and UDP traffic to evaluate the effectiveness of SDN routing algorithm under various network load conditions. The experimental configurations are presented in Table 1. The experimental setup in this paper is based on the work of Shirmarz [18], Gunavathie [20], and others, as depicted in Table 2.

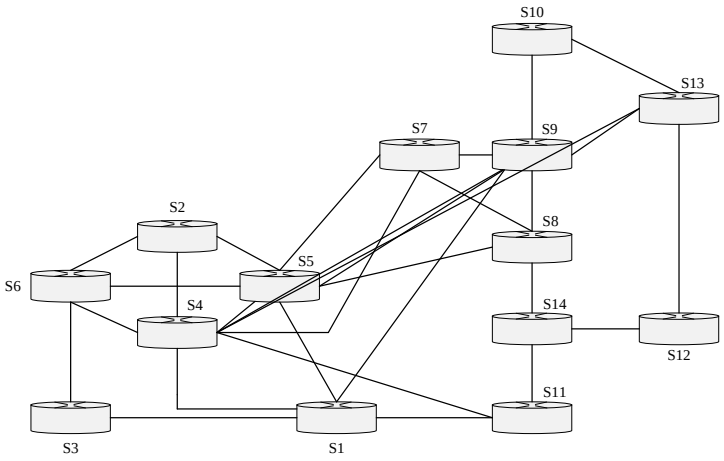

FIGURE 9. NYC Topology

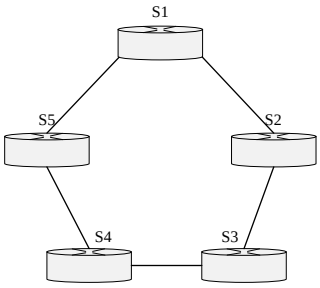

FIGURE 10. Ring Topology

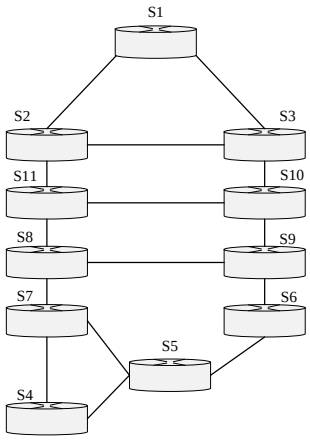

FIGURE 11. Abilene Topology

TABLE 1. Configuration of the experiment

Entry	Configuration Information
OS	Linux Ubuntu
CPU	12th Gen Intel i7
Memory	16GB
Programming Language	Python
Simulation Platform	Mininet
Controller	Ryu

TABLE 2. Experimental environment

Topology	Nodes	Links	Max BW	Min BW	Max Delay	Min Delay	Max LT	Min LT	Flows	
			(Mbps)	(Mbps)	(ms)	(ms)	(%)	(%)	Mice	Elephant
Ring	5	5	100	10	120	10	7	1	5	3
NYC	14	28	100	10	120	10	7	1	15	10
Ring	11	15	100	10	120	10	7	1	15	10

3.2. **Training phase.** Training strategy: Refer to the training optimization process of Gunavathie et al [20]. First, we pre-trained the GRU model with attention layer with four different combinations of hyperparameters and trained the combination with better performance more times to obtain the final trained model. The model is trained using the generated sequence of indicators as input to predict future indicators. The generated dataset has 50,000 data points, including delay, loss tolerance and bandwidth. The evaluation indicator is mean squared error (MSE).

Training process: First, we pre-train the model with four different hyperparameter combinations, mainly adjusting the learning rate (LR), the hidden parameters of the RNN (RNNH), and the number of RNN layers (RNNL), as shown in Table 3. Eighty percent of the data used for training and the rest is used to validation, batch_size refers to the number of links. We use these four different hyperparameter combinations for 25 epochs training, as shown in Figures 12 and 13. The purpose is to find the optimal hyperparameter combination.

TABLE 3. Combination of hyperparameters

Hyperparameter combination	LR	RNNH	RNNL
Combination 1	0.0001	1024	1
Combination 2	0.001	1024	1
Combination 3	0.001	512	2
Combination 4	0.01	1024	1

It can be observed from Figure 12 that after training for 25 epochs, the training loss of combination 2 is smaller. In addition, it can be observed that the training losses of combination 1 and 3 are comparable, but the learning rate of combination 1 is 0.0001, which is slower than the learning rate of combination 3, which is 0.001. In summary, we further train combinations 2 and 3 to obtain the optimal hyperparameter combination.

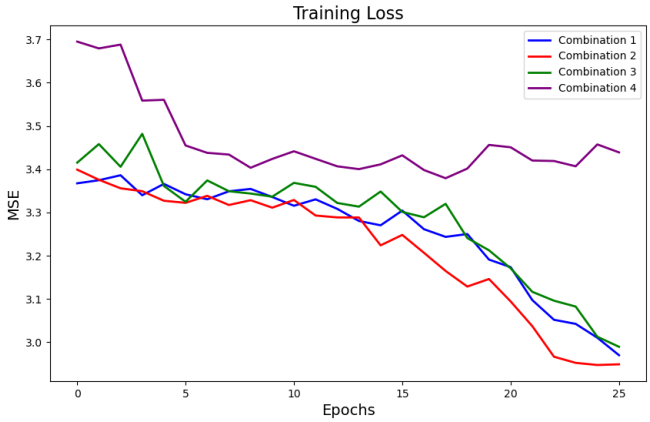

FIGURE 12. Initial hyperparameter combination training loss

We continued to train combinations 2 and 3 for 250 epochs. As shown in Figure 14, we can observe that after the 150-th epoch, the training loss of combination 2 is significantly less than that of combination 3. Therefore, we terminated the training of combination 3 at the 150-th epoch and found combination 2 to be the optimal hyperparameter combination. However, it is observed that the loss of combination 2 fluctuates greatly in the first 50 epochs, and the loss fluctuates less between 50 and 150 epochs, and the loss fluctuates

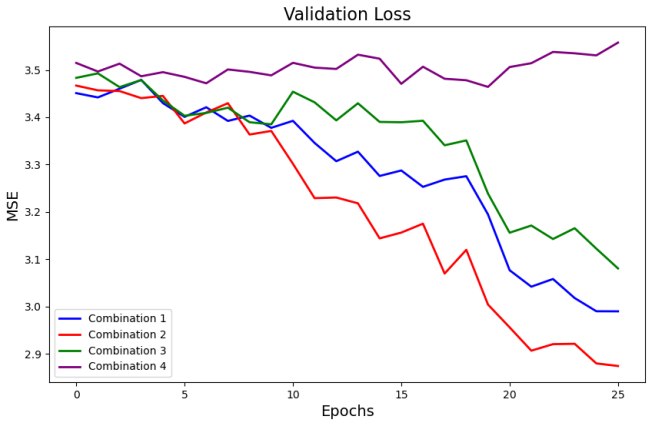

FIGURE 13. Initial hyperparameter combination validation loss

even less after the 150-th epoch. The loss value fluctuation is caused by the learning rate. To solve this problem, we adjusted the learning rate. The specific adjustment strategy is to use 0.001 as the learning rate for training in the 0-th to 50-th epoch, 0.0001 as the learning rate for training in the 50-th to 150-th epoch, and 0.00001 as the learning rate for training in the 150-th to 250-th epoch. As shown in Figures 14 and 15, we observe that for combination 2, after using learning rate adjustment strategy, the model performance is significantly better than before, and as the number of epochs raises, the training loss steadily declines.

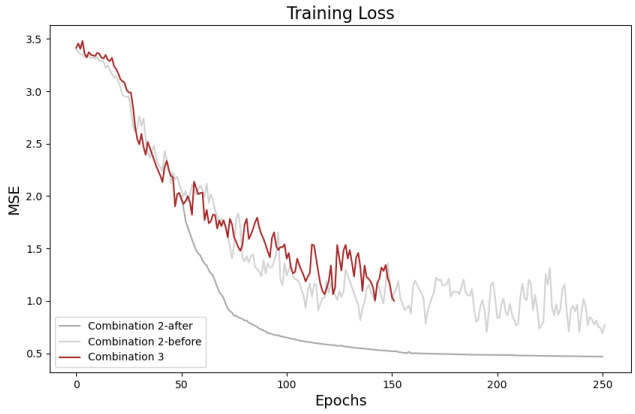

FIGURE 14. The impact of learning rate adjustment strategy on training loss

The trained model has been converted into an Open Neural Network Exchange (ONNX) file and deployed in SDN for inference. After normalization, the data was fed into ONNX and loaded. The output size is (N, number of parameters) and is used to map the weight of each link to make path decisions.

3.3. **Experimental Results and Data Analysis.** The decision model utilizes the predicted indicators to determine the link weights and make decisions. The weight is between 0 and 1. The link combination with the smallest weight will be considered for transmission. The model updates the link weights every 5s, and thinking the indicators of the previous 5 sequences, the GRU model with an attention layer determines link weights at time $t + 1$. Experiments are conducted on NYC, Ring, and Abilene topologies. The experimental simulation runs for 15 minutes (900s) and calculates the average delay, loss

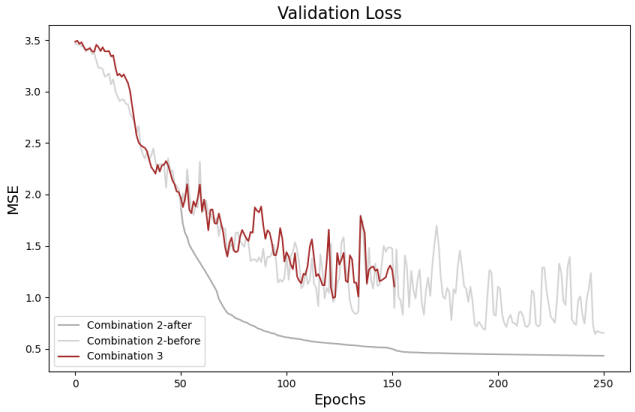

FIGURE 15. The impact of learning rate adjustment strategy on validation loss

tolerance, and bandwidth. Implemented in Mininet is the Dijkstra algorithm for comparison purposes. Utilizing the experimental setup outlined in Table 3, experiments were conducted on three topologies.

3.3.1. *Comparative Experiment.* Default shortest path algorithm and the decision model of this paper have been implemented in the experimental environment of Table 2 and tested in three different topologies. The results are shown in Table 4. From Table 4, we

TABLE 4. Comparison of the proposed work with default shortest path implementation

Name	This Paper			Dijkstra Algorithm		
	Delay (Millisecond)	Loss Tolerance (%)	Throughput (bytes per second)	Delay (Millisecond)	Loss Tolerance (%)	Throughput (bytes per second)
Ring	12.337	1.2	1977069.854	12.421	1.3	1873945.371
NYC	20.952	1.9	2437415.172	21.568	2.2	2171643.374
Abilene	18.514	1.7	2287032.075	19.739	1.9	2011543.272

can see that our model demonstrates significant superiority over the default shortest path algorithm regarding network indicators such as delay, loss tolerance, and throughput. In the Ring topology, the two algorithms perform comparably, with our decision model having slightly less delay. Since the Ring topology only contains 5 nodes, has fewer links and a simple structure, the decision model has little effect on improving the Ring topology. Considerable improvements in delay and loss tolerance can be observed in both NYC and Abilene. NYC and Abilene have relatively complex topologies with many links, and they show better performance in complex network topologies compared to default shortest path algorithm. Our decision model achieved a 0.6% reduction in delay compared to default shortest path in the Ring, 2.8% in the NYC, and 6.2% in the Abilene. Loss tolerance decreased by 7.7% in the Ring, 13.6% in the NYC, and 10.5% in the Abilene. Throughput saw an increase of 5.5% in the Ring, 12.2% in the NYC, and 13.7% in the Abilene.Significant improvements across these three network indicators are demonstrated by the experimental results of the proposed model. This paper also compares with some proposed path decision models, mainly including an algorithm called TOPSIS proposed by Shirmarz et al. to determine optimal path for traffic and TAOR algorithm proposed by M.A. Gunavathie et al. The experimental environment of this paper is built with reference to these two studies, and the comparison data comes from the TAOR algorithm. As shown in Tables 5 and 6, we mainly compare the NYC topology, comparing the reduction percentage of delay and loss tolerance, and the increase percentage of throughput.

Table 5. Comparison of TOPSIS and This Paper

Network Indicators	TOPSIS [18]			This Paper (NYC topology)		
	Dijkstra Algorithm	TOPSIS	% decrease	Dijkstra Algorithm	This Paper	% decrease
Average delay (seconds)	0.0330	0.0480	-45.5	0.0216	0.0210	2.8
Loss Tolerance (%)	4.4	3.6	18.1	2.2	1.9	13.6
Throughput (bytes per second)	/	/	/	2171643 .374	2437415 .172	% increase 12.2

Table 6. Comparison of TAOR and This Paper

Network Indicators	TAOR [20] (NYC topology)			This Paper (NYC topology)		
	Dijkstra Algorithm	TOPSIS	% decrease	Dijkstra Algorithm	This Paper	% decrease
Average delay (seconds)	0.0059	0.0058	1.7	0.0216	0.0210	2.8
Loss Tolerance (%)	/	/	/	2.2	1.9	13.6
Throughput (bytes per second)	2271830 .082	2880833 .345	% increase 26.8	2171643 .374	2437415 .172	% increase 12.2

As Tables 5 and 6 illustrates, in comparison with TORA, the delay of the proposed model is higher than that of TORA. The reason is that the TORA algorithm focuses on improving delay and jitter, especially achieving significant results in improving delay. The TOPSIS algorithm focuses on improving path utilization, and the path utilization of this algorithm shows a 59% improvement. Through utilizing predictive analytics and dynamically adjusting link weights, our algorithm enhances network performance, enabling real-time adaptation to traffic changes. Comprehensive consideration is given to various indicators that affect network performance, with the focus on enhancing the overall performance of the network, including delay, loss tolerance and bandwidth, and boosting the overall service quality of the network. Although the TOPSIS performs well in terms of path utilization and the TORA performs well in terms of delay improvement, our method comprehensively considers the indicators that affect the network and better improves network reliability.

3.3.2. *Ablation Experiment.* We introduced an attention layer in the hidden layer of GRU to consider the importance of indicator information at different times to enhance the effectiveness of the overall decision model. The performance comparison of the decision model before and after the introduction of the attention layer is shown in Table 7. It's evident from Table 7 that the performance enhancement of the decision model in the Ring is small before and after the attention mechanism was introduced. Due to the simple structure of the Ring topology, original model without the introduction of the attention layer can achieve good performance. However, in the Ring and Abilene topologies, due to the complexity of the Ring and Abilene network topologies, the connections and information transmission between nodes are more complex, so the decision model is needed to capture the importance of network indicators at different times to the model prediction,

TABLE 7. Results of ablation experiment

Name	Before			After		
	Delay (Millisecond)	Loss Tolerance (%)	Throughput (bytes per second)	Delay (Millisecond)	Loss Tolerance (%)	Throughput (bytes per second)
Ring	12.379	1.2	1965612.269	12.337	1.2	1977069.854
NYC	21.229	2.1	2295337.049	20.952	1.9	2437415.172
Abilene	18.927	1.8	2157073.754	18.514	1.7	2287032.075

and achieve more precise predictions in more complex network topologies. This has a great impact on the determination and update of link weights and the final path decision.

3.4. **Engineering Applications.** The growth of mobile data has promoted the development of fifth-generation (5G) and sixth-generation (6G) mobile networks. The data flow in mobile communication networks is soaring, and early mobile networks cannot meet the needs of users [26]. In 2024, more than 80% of global mobile data traffic will come from video traffic, with 13 billion mobile devices connected to the Internet. Since video transmission requires high bandwidth and low delay, how to efficiently transmit video streams through the network to provide users with high-quality video services has become a pressing issue to be addressed. As shown in Figure 16. Our proposed decision algorithm can be combined with the video segmentation algorithm to significantly improve the video service quality, especially in poor network environments [27].

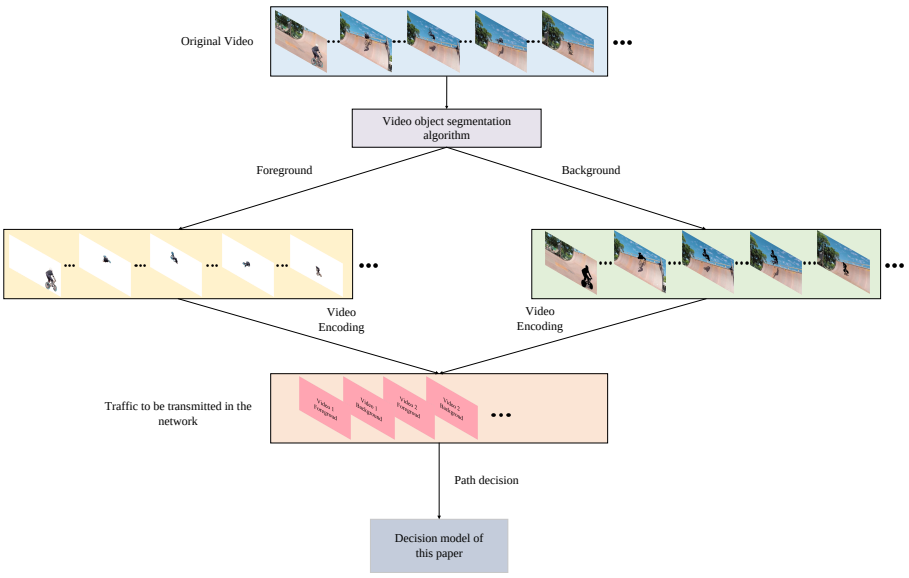

FIGURE 16. The practical application of the algorithm presented in this paper

4. **Conclusion.** The primary contribution of this study is to combine GRU with SDN and introduce attention to enhance the effectiveness of path decision model. Simultaneously, we comprehensively considered key network indicators and achieved comprehensive consideration of indicators through methods such as normalization and fusion. We conducted experiments on three topologies and the proposed model significantly improved network effectiveness, as indicated by the results. Compared with default shortest path,

our decision model reduces the delay by 0.6%, 2.8%, and 6.2% on the three topologies, reduces the loss tolerance by 7.7%, 13.6%, and 10.5%, and improves the throughput by 5.5%, 12.2%, and 13.7%, respectively. Compared with existing path decision algorithms, our approach comprehensively considers network indicators, prioritizes the improvement of comprehensive service quality, and improves network reliability. Finally, we conduct ablation experiments to verify the influence of attention on the effectiveness of decision model. Our model not only improves the accuracy and reliability of routing decisions, but also provides a new idea for SDN optimization.

REFERENCES

[1] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: an enhanced authentication protocol for iot-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, no. 1, p. 36, 2023.

[2] T.-Y. Wu, L. Wang, X. Guo, Y.-C. Chen, and S.-C. Chu, "Sakap: Sgx-based authentication key agreement protocol in iot-enabled cloud computing," *IEEE Systems Journal*, vol. 14, no. 17, p. 11054, 2022.

[3] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in sdn-based data center: a survey," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7545–7593, 2020.

[4] M. He, A. , Varasteh, and W. Kellerer, "Toward a flexible design of sdn dynamic control plane: An online optimization approach," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1694–1708, 2019.

[5] H. Yao, P. Zhang, J. Wang, C. Jiang, and M. Guizani, "Machine learning aided load balance routing scheme considering queue utilization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7987–7999, 2019.

[6] L. Huang, M. Ye, Y. Wang, H. Qiu, and X. Deng, "Intelligent routing method based on dueling dqn reinforcement learning and network traffic state prediction in sdn," *Wireless Networks*, vol. 30, pp. 4507–4525, 2022.

[7] X. Zhou, M. Su, Z. Liu, Y. Hu, B. Sun, and F. Guanghui, "Smart tour route planning algorithm based on naïve bayes interest data mining machine learning," *ISPRS International Journal of Geo-Information*, vol. 9, no. 2, p. 112, 2020.

[8] G. Kim, Y. Kim, and H. Lim, "Deep reinforcement learning-based routing on software-defined networks," *IEEE Access*, vol. 10, pp. 18 121–18 133, 2022.

[9] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, and S.-C. Tsai, "Rl-routing: An sdn routing algorithm based on deep reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3185–3199, 2020.

[10] Y. Zhang, L. Qiu, Y. Xu, and S. Wang, "Multi-path routing algorithm based on deep reinforcement learning for sdn," *Applied Sciences*, vol. 13, no. 22, p. 12520, 2023.

[11] T. Zhu, X. Chen, L. Chen, and W. Wang, "Gclr: Gnn-based cross layer optimization for multipath tcp by routing," *IEEE Access*, vol. 8, pp. 17 060–17 070, 2020.

[12] A. Swaminathan, M. Chaba, D. K. Sharma, and U. Ghosh, "Graphnet: Graph neural networks for routing optimization in software defined networks," *Computer Communications*, vol. 178, no. 1, pp. 169–182, 2021.

[13] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[14] B. Yan, Q. Liu, J. Shen, and D. Liang, "Flowlet-level multipath routing based on graph neural network in openflow-based sdn," *Future Generation Computer Systems*, vol. 134, pp. 140–153, 2022.

[15] X. Zheng, W. Huang, H. Li, and G. Li, "Research on generalized intelligent routing technology based on graph neural network," *Electronics*, vol. 11, no. 18, p. 2592, 2022.

[16] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv-prints*, p. arXiv:1406.1078, 2014.

[17] M. A. Gunavathie and S. Umamaheswari, "Bbr-a novel bandwidth based routing algorithm in software defined networking," *2022 1st International Conference on Computational Science and Technology (ICCST)*, pp. 1038–1041, 2022.

[18] A. Shirmarz and A. Ghaffari, "Automatic software defined network (sdn) performance management using topsis decision-making algorithm," *2022 1st International Conference on Computational Science and Technology (ICCST)*, vol. 19, no. 16, 2021.

[19] R. Mohammadi and R. Javidan, "Research on generalized intelligent routing technology based on graph neural network," *Journal of Reliable Intelligent Environments*, vol. 8, pp. 247–260, 2022.

[20] M. Gunavathie and S. Umamaheswari, "Traffic-aware optimal routing in software defined networks by predicting traffic using neural network," *Expert Systems with Applications*, vol. 239, no. 1, p. 122415, 2024.

[21] B. Liu, W. Zhou, H. Ren, and Z. Fan, "Intelligent prediction and data service ptimization scheme of power grid based on improved bee colony algorithm," *Journal of Network Intelligence*, vol. 9, no. 1, pp. 253–272, 2023.

[22] X. Zhang, Y. Wei, and Z. Hashim, "Improvement of swarm intelligence algorithm and its application in logistics network routing," *Journal of Network Intelligence*, vol. 8, no. 4, pp. 1077–1094, 2023.

[23] N. Feng, T.-Y. Wu, and Y. Liang, "A deep dynamic neural network model and its application for ecg classification," *Journal of intelligent*, vol. 43, no. 2, 2022.

[24] D. Bahdanau, K. Cho, and Y. Bengio, "A deep dynamic neural network model and its application for ecg classification," *arXiv-prints*, p. arXiv:1406.1078, 2014.

[25] "New york metro ibx data center data sheet," *Available: https://www.equinix.com/resources/datasheets/nyc-metro-data-sheet*, 2020.

[26] L. Yang, Y.-C. Chen, and T.-Y. Wu, "Provably secure client-server key management scheme in 5g networks," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, 2021.

[27] C.-M. Chen, Q. Miao, S. Kumar, and T.-Y. Wu, "Provably secure client-server key management scheme in 5g networks," *Transactions on Emerging Telecommunications Technologes*, vol. 34, no. 11, p. e4751, 2023.