DDPG-SDPCR: A DDPG-based Software Defined Perimeter Components Redeployment

Zheng Zhang[1], Quan Ren[2], Jie Lu[2], Yuxiang Hu[2*] and Hongchang Chen[2]
[1] Institute of Information Technology, PLA Strategic Support Force Information Engineering University, Zhengzhou 450000 China
[e-mail: 975644214@qq.com]
[2] Institute of Information Technology, PLA Strategic Support Force Information Engineering University Jianxue Street 7, Zhengzhou, Henan, China
[e-mail: huyuxiangchn@163.com]
[*]Corresponding author: Yuxiang Hu

Received March 11, 2024; revised July 29, 2024; accepted August 19, 2024; published September 30, 2024

Abstract

In wide area SDP networks, the failure of SDP components caused by malicious attacks will be accompanied by different deployment locations, profoundly affecting network service latency. However, traditional deployment methods based on prior knowledge are no longer applicable to dynamic SDP networks. This article proposes a dynamic and dimensionally variable deployment mechanism DDPG-SDPCR for SDP components based on DDPG, which enhances the network's endogenous security capability and improves attack tolerance. Based on this, we constructed corresponding mathematical models for latency, load balancing, and attack tolerance. The DDPG-SDPCR mechanism dynamically deploys new SDP nodes to replace faulty nodes based on the real-time status of the network, thereby achieving imperceptible attack tolerance for users. We have implemented a wide area SDP prototype with endogenous security capabilities and evaluated it under different network topologies, traffic sizes, and network attacks. The evaluation results indicate that under high traffic conditions, our proposed redeployment mechanism outperforms the baseline by 36.42% in latency, and only increases by 19.24% compared to the non attacked situation.

Keywords: SDP, DDPG, Attack tolerance, Endogenous security

This research was supported by a research grant from the Science and Technology Innovation leading Talents Subsidy Project of Central Plains [Grant No. 244200510038]. We express our thanks to Prof. Hu who checked our manuscript.

http://doi.org/10.3837/tiis.2024.09.014 **ISSN : 1976-7277**

1. Introduction

Software Defined Perimeter (SDP) [1] is a zero trust architecture [2], which provides enterprises with dynamic and flexible network security logical perimeter to isolate resources such as applications and services on insecure networks with unknown security threats. SDP is first defined by the CSA in 2014 as an abstraction of network perimeter from physical implementation to logical definition. The emergence of this model is in response to the surge in network devices and the increasing mobility of devices, leading to the gradual melting of traditional network boundaries. The SDP architecture provides a new type of security model that provides better protection by verifying the identity of resource visitors to prevent threats both inside and outside the boundary. By hiding resources from unauthorized network entities, SDP can effectively eliminate some malicious attacks, including violent attacks, network traffic attacks, and other attacks on [3-4]. The SDP architecture typically includes SDP components such as an SDP controller, SDP gateway, initiator host, and receiver host in hardware. The software includes single packet authentication algorithms, encryption algorithms, etc. In SDP networks, the principle of minimum privilege is implemented to encrypt network traffic so that all resources can be safely accessed without crossing boundaries, regardless of the user's location. Through an isolated, on-demand, and dynamically configured trusted logic layer, the SDP architecture can provide secure isolation and protection for internal network resources against malicious behavior by non authenticated users.

However, in the face of malicious attacks targeting SDP components themselves, such as malicious tampering, vulnerability backdoor attacks, etc., once the malicious attacks lead to failure, a centralized SDP architecture will not be able to meet the normal security function services. Although the distributed SDP architecture [5] can avoid lightweight cyber attacks to a certain extent, it will still reduce its processing performance and increase the communication delay. With the increase of malicious attacks, the processing performance of SDP will further decline, and the security protection function provided by SDP will eventually fail. Therefore, how to improve the attack tolerance of SDP architecture and avoid single point of failure on the premise of ensuring the service function and communication delay has become the key of research.

The SDP component which is logically centralized but physically distributed is a feasible solution. In SDP multi-component network architecture, the number and location of SDP components have a great impact on network performance, and its placement has become a challenge in the current research. Placement issues are not specific to the SDP architecture alone. This problem is also raised in SDN and Network Function Virtualization (NFV), that is, to find the most suitable deployment location of distributed SDN controller and resources of Virtualized Network Function (VNF) [6, 7]. In addition, to find the most suitable node in Named Data Networking (NDN) [8] architecture to play the role of NDN controller [9]. These problems have some common points, but also have their unique places. In principle, the deployment optimization of SDP components in Wide Area Network (WAN) means obtaining the number and placement location of SDP components, with the goal of optimizing such as minimizing delay [10, 11, 12], balancing load [13, 14, 15], and reducing energy consumption [16, 17]. In the previous methods, greedy or heuristic methods are used, and load balancing is achieved through one of the above three goals. However, existing traditional methods have not taken into account the special problems faced by SDP component deployment. The ultimate goal of deployment is to enable users to tolerate attacks without perception, rather than simply optimizing latency or energy consumption. With the development of deep reinforcement learning, algorithms based on Deep-Q-Network (DQN) have reduced the action

dimension of agent output, enabling DQN to better solve problems that require multi-dimensional action output. However, when there is excessive load in the wide area SDP network or some SDP components suffer from network attack failures, it is necessary to temporarily add network equipment to ensure the normal service of users. In this case, the number of SDP components or switches may change. However, algorithms based on DQN limit the number of network nodes to a fixed value; Therefore, it cannot adapt to the dynamic changes in the number of network devices in malicious attack scenarios.

In order to meet the requirements of processing speed and multiple optimization objectives for SDP component deployment, this paper proposes a dynamic and dimensionally variable optimization model based on Deep Deterministic Policy Gradient (DDPG-SDPCR). The model considers both minimum latency and attack tolerance, and balances traffic load. Deep Reinforcement Learning (DRL) allows network entities to learn and establish knowledge about network states, and to solve problems with large state and action spaces [18], achieving autonomous and efficient decision-making capabilities. Here, we proposed a wide area SDP architecture that has the ability to redeploy SDP components while considering potential network attack threats. We considered the data flow delay, link bandwidth and load in the network, as well as tolerance requirements, and designed a mechanism to consider the incremental deployment of SDP components in the scenario of SDP component failure after being attacked during user traffic forwarding. Unlike the node deployment scheme based on DQN, our proposed DDPG algorithm solves the constructed model and uses bit masks to adjust the effective dimensionality of actions in the DDPG algorithm, allowing the algorithm to operate when the number of deployed SDP components is variable. Specifically, DDPG-SDPCR allows the dimensions of deployment nodes to be dynamically adjusted based on the decisions of network administrators, which means that the goal is to increase attack tolerance and reduce deployment costs as much as possible based on user insensitivity or reduced deployment increments. In addition, based on reinforcement learning algorithms, DDPG-SDPCR does not require a large number of training samples and can obtain new data through exploration of the environment, and use the new data to repeatedly update and iterate the existing model. In the algorithm proposed in this article, random noise variables are also added to increase the exploration range to prevent getting stuck in local optima.

The primary contributions of this study are summarized below.

(1) We propose a wide area SDP framework with endogenous security capabilities, which can provide users with secure access to remote intranet resources over a wide area. In addition, considering potential network attacks, it enables to redeploy SDP components and meet user latency and bandwidth requirements after some SDP components fail due to malicious attacks.

(2) We propose a dynamic and dimensionally variable SDP component redeployment mechanism DDPG-SDPCR, which can detect and remove failed SDP components after some SDP components fail due to malicious attacks based on consistency verification. Then, based on the redeployment algorithm, considering constraints such as network traffic, link bandwidth, and latency, suitable nodes in the network are selected as deployment locations for the new SDP components.The source code of DDPG-SDPCR implementation and the related algorithms will been released at Github.

(3) We propose a wide area SDP prototype with endogenous security capabilities. Simulation results show that the redeployment of components in the wide area SDP architecture based on DDPG and bit mask algorithms outperforms the baseline by at least 11.56% in terms of latency, link load, and attack tolerance.

The rest of this article is organized as follows: related work can be found in Section 2. The problem statement and mathematical modeling will be explained in the Section 3. In Section

4, we provided a detailed explanation of the proposed algorithm and corresponding parameters. The evaluation results of the model are presented in Section 5. Finally, Section 6 concludes this study.

2. Related Works

This section provides an overview of the relevant work. Firstly, relevant work on potential mathematical problems is provided, followed by an introduction to the application of DRL in network edge or cloud placement problems. Finally, explanation is given on the universality and specificity of SDP component deployment problem.

Heller et al. [19] first proposed the control plane deployment problem by analyzing the impact of controller placement on the average delay and maximum delay of the network. Since then, people have done a lot of work on it. In real life, the center of logistics network chooses the best location when the factory, warehouse or other equipment in a given network topology. Therefore, this problem is also known as the location problem of factories, facilities or warehouses. Generally, people use mixed integer linear programming, such as the software based on IBM decision optimization modeling for python [20] to model and solve it. If the optimization objective is the average value (delay or hops and other optimization objectives), the problem is classified as a k-means problem [21]. If the optimization goal is the maximum value, the problem is classified as a k-center problem [22]. The difference between the two is that the selected nodes are the average or random values in the cluster. Generally, the latter has a larger amount of calculation, while the former has the problem that the output result may not be in the set. Heller's work [19] provides a further discussion on this general problem. In [23] and [24], different aspects and different methods of facility location issues are also outlined, focusing on "uncertainty", such as uncertain flow demand or delay. In addition, there are deployment schemes based on density clustering [25] and heuristic algorithms for linear programming [26, 27, 28]. However, due to various constraints and heterogeneous variables in controller deployment, the optimization problem is NP hard. Moreover, these works focus on general and universal theoretical issues, which do not address the specific issues of device deployment in wide area networks, which involve multiple parameters.

The application of DRL in the field of networks provides effective tools and new solutions for responding to dynamic changes in networks, optimizing the deployment of network devices and services. Intelligent agents can iteratively interact with random environments to find the optimal solution. Reza et al. [29] proposed a controller placement and allocation strategy based on reinforcement learning method, which includes a wide range of constraints, including elasticity, delay, load balancing and controller capacity. The author explained that they first proposed a machine learning method to solve the placement and allocation problem of multi-constraint elastic controllers. However, due to the traditional reinforcement learning (RL) method, the strategy has the problems of long learning time and slow convergence speed, which is difficult to be applied to large networks or delay sensitive networks. Alejandro et al. [30] proposed a VNF deployment mechanism based on DQN across single board computers (SBC) clusters. This mechanism selects the most appropriate node in multiple clusters, and deploys VNF according to the node's resource and event requirements to optimize the energy consumption in SBC. This research mainly considers resource cost and energy consumption, and does not optimize network indicators such as delay and load balancing. Li et al. [31] proposed an algorithm based on multi-agent DQN (MADQN) to realize the dynamic placement of controllers. The algorithm allows the control plane to change dynamically with the change of flow. In addition, the algorithm based on MADQN reduces the action dimension

of agent output, so that DQN can better solve the problem of multi-dimensional action output. However, the MADQN based algorithm uses DRL to deploy dynamic controllers. When the network load in a hot area is too large or some network equipment is disabled by network attack, network facilities (such as network access points) must be temporarily added to provide sufficient network service capacity. In this case, the number of network devices, such as switches, will change. However, the MADQN based algorithm limits the number of switches to a fixed value. Therefore, it cannot adapt to the application scenario of dynamic changes in the number of network devices. Yu et al. [32] proposed an edge computing and caching solution based on DDPG. The solution considers the diversity of services requested by users and the dynamic communication conditions between Multi-access Edge Computing (MEC) server and users, so as to jointly optimize task scheduling and resource allocation in continuous action space. The simulation results show that the DRL inspired resource allocation scheme is obviously superior to other comparison strategies, and achieves the optimal resource allocation scheme. Unfortunately, although the above literature considers a variety of network parameters and constraints, it does not study and design the faults caused by cyber attacks.

Table 1. Comparison of existing methods.

	Delay optimization	Load balancing optimization	Energy consumption optimization	Attack tolerance	Comment
Clustering [21,22,25]	√	×	×	×	Single optimization objective
Heuristic [26,27,28]	√	×	√	×	Large calculation and time consumption
DQN [29,30,31]	√	√	×	√	Fixed vector dimension

For the deployment of network equipment or service functions in SDN, the above research considers network parameters or user requirements such as delay and load, so that the control plane can meet various constraints and achieve load balance between different controllers. Some of the methods proposed in the research even take into account the dynamic changes of the network, so that its deployment scheme can be dynamically adjusted according to the changes of the network state. However, the above research on equipment or service placement for SDN and SDP component deployment has a relationship of universality and particularity. The particularity of SDP component deployment is to consider the function failure caused by malicious attacks and minimize the impact. The above research does not consider the impact of network attack, hence its deployment scheme cannot adapt to the real environment faced by SDP wide area network.

3. Problem Statement and Mathematical Modeling

As mentioned earlier, the way to solve the SDP component deployment problem is to obtain the number and location of SDP components to be deployed, so that they have the lowest cost and delay, as well as a certain degree of attack tolerance. In this section, we first introduce the wide area SDP architecture. Then, we introduce the delay, load balancing and tolerance model of SDP component deployment.

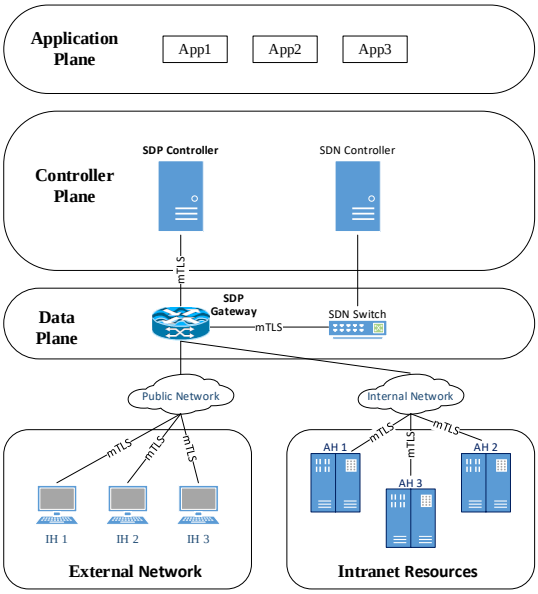

Fig. 1. The structure of wide area SDP network.

3.1 Wide Area SDP architecture

Fig. 1 shows the wide area SDP network structure. Based on SDN structure, the structure is divided into three layers: data plane, control plane and application plane. The data plane includes mobile devices, edge servers, cloud computing centers, switches and other network devices. User equipment can access the edge SDP gateway through the access point to obtain SDP controller authentication, and then obtain computing or data storage services. The edge server and cloud center use wired network connection. The network services of devices in the data plane are managed through the control plane. The application layer provides resource management, uninstall services and other services.

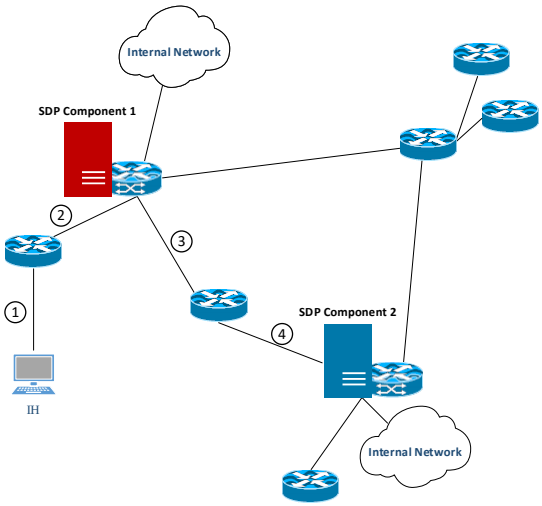

Fig. 2. Solution 1 under attack.

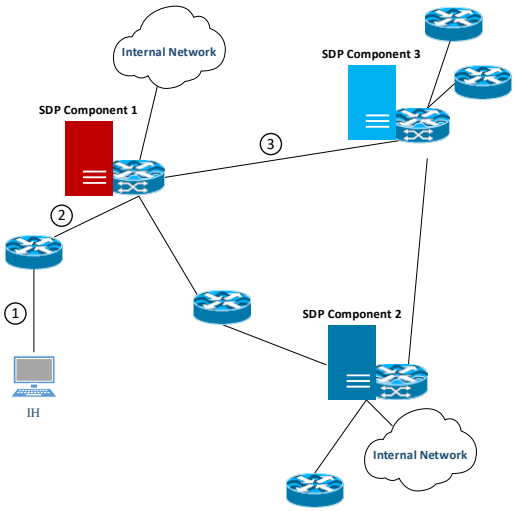

Fig. 3. Solution 2 under attack.

Fig. 2 and **Fig. 3** show two solutions when SDP components are attacked and fail. Red is the failed SDP component, dark blue is the normal SDP component, and sky blue is the newly deployed SDP component. **Fig. 2** shows the redirection strategy based on greedy algorithm, that is, when SDP component 1 is attacked and fails, the communication path from the initiating host to the intranet resources is redirected from ① → ② to ① → ② → ③ → ④. In this way, the communication path bypasses the attacked SDP component at the cost of increasing the delay and the load of SDP component 2. **Fig. 3** shows the redeployment strategy based on the DDPG-SDPCR algorithm proposed in this study, that is, when SDP component 1 is attacked and fails, the new SDP component 3 is redeployed by analyzing the current network state. The communication path from the initiating host to the intranet resources is changed from ① → ② to ① → ② → ③. In this way, the destination SDP node is replaced while reducing the additional delay as much as possible.

3.2 Mathematical Model

In this study, we use a directed acyclic graph with variable nodes to represent the network topology. Here, the maximum number of nodes is n. Each node is a switch, and SDP components can be deployed on any node. Graph $\mathrm{G} = (\mathrm{Q};\ \mathrm{E})$shows the network topology, where q is the set of switches $\mathrm{Q} = \{\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{N}\}$, and E is the set of edges between nodes. The number of SDP components in the network is m, and the SDP component set is marked as $\mathrm{G} = \{\mathrm{g}_1, \mathrm{g}_2, \ldots, \mathrm{g}_\mathrm{M}\}$. In addition, we set a binary variable $X_{i,j}$ to represent the connection between node i and node j.

$$X_{i,j} = \begin{cases} 0, node\ i\ is\ not\ connetced\ with\ node\ j; \\ 1, node\ i\ is\ connetced\ with\ node\ j. \end{cases} \tag{1}$$

3.2.1 Delay Model

SDP architecture provides secure and stable identity authentication and connection establishment services for all kinds of users, in which there are many low latency applications. Therefore, SDP must process network requests under certain delay constraints. SDP covers a wide area. In addition, some nodes are far away and the network load distribution is uneven. These characteristics lead to different data forwarding times of switches and different processing times of SDP components for network requests. Therefore, the component deployment in SDP architecture must consider propagation delay and transmission delay.

1. Propagation delay

As mentioned above, in the wide area SDP network, some nodes are far away from each other, causing a large propagation delay in these nodes. Therefore, the centralized SDP component cannot handle the network requests from these devices in time. The propagation delay between nodes and the distance between nodes are related to the signal transmission speed of the link between nodes. Therefore, the propagation delay between node i and node j is expressed as follows:

$$t_{i,j}^{p} = \frac{X_{i,j} \cdot d_{i,j}}{v_{i,j}} \tag{2}$$

$d_{i,j}$ is the distance between node i and node j, and $v_{i,j}$ is the propagation speed between node i and node j.

2. Transmission delay

In the wide area SDP network, wired connections are used between switches and SDP gateways, between switches and switches, and between SDP controllers and SDP gateways. In the wired network, due to the stability of the network environment, the interference between different devices can be ignored. Therefore, the transmission delay between node i and node j is defined as follows:

$$t_{i,j}^{t} = \frac{X_{i,j} \cdot V_{i,j}}{B_{i,j}} \tag{3}$$

$V_{i,j}$ is the amount of data between node i and node j, and $B_{i,j}$ is the link bandwidth between node i and node j. Therefore, the complete link delay from node i to node j is defined as:

$$t_{i,j} = t_{i,j}^{p} + t_{i,j}^{t} \tag{4}$$

For SDP component node i, we define its node delay t_i as the sum of the delays experienced by the network request flow with it as the access point, namely:

$$t_i = \sum_{j=1}^{J} t_{i,j} \tag{5}$$

where, J is the number of network request flows with node i as the access end.

We take the maximum delay of all SDP component nodes as t_{max},

$$t_{max} = \max_{1 \le i \le M} \{t_i\} \tag{6}$$

3.2.2 Link Bandwidth Utilization Limitation and Load Balancing Model

In SDN network, bandwidth is a limited system resource on which control flow and data flow are transmitted. Therefore, only a limited amount of traffic can be transmitted in each link. In addition, when the link is overloaded, the network performance will decline sharply, and the packet loss rate of data transmission will also increase sharply. In addition, the effective throughput of the network decreases, and the total delay caused by data retransmission will also increase. In extreme cases, link congestion will result in local deadlocks that cannot be used. Therefore, in order to make full use of bandwidth resources and ensure that the link is not overloaded, the bandwidth utilization of the link must be controlled within a reasonable range.

The bandwidth utilization of link k between two directly connected nodes can be defined as follows:

$$\eta^{k} = \frac{f^{k}}{B^{k}} \tag{7}$$

where, f^{k} is the traffic size of link k and B^{k} is the bandwidth of link k.

Due to the difference of path planning and initial access point location, the traffic size on each link will be different. Here, we use the standard deviation of traffic to measure the load difference between SDP components. The load size of all links of an SDP component, the average value and standard deviation of the load of all SDP components are defined as follows:

$$\phi_j = \frac{\sum_{k=1}^{K_j} f^k}{K_j} \tag{8}$$

$$\phi_{avg} = \frac{\sum_{j=1}^{M} \phi_j}{M} \tag{9}$$

$$\Delta\phi = \sqrt{\frac{\sum_{j=1}^{M} (\phi_j - \phi_{avg})^2}{M}} \tag{10}$$

where, K_j is the total number of requests received by SDP components g_j. The smaller the standard deviation $\Delta\phi$, the more balanced the load of SDP gateway in the network.

3.2.3 SDP Component Attack Tolerance Model

In order to ensure the continuous and stable operation of SDP architecture, it is necessary to maintain sufficient attack tolerance on the network link between SDP components and protected network resources. SDP components have high attack tolerance and can continuously maintain links with internal network resources and external access users in case of malicious attacks. In the SDP architecture with SDN enabled, the control flow and data flow use the same link to transmit data. Therefore, the attack tolerance of SDP components is related to (i) the recovery time of cleaning and restarting after SDP components are attacked and failed, that is, the node tolerance; and (ii) the tolerance of SDP component deployed node of all links.

1. Node tolerance and deployment cost

Based on the types of SDP components, the recovery time of cleaning and restarting is different. In addition to human factors, we generally believe that the more expensive SDP components are, the shorter the recovery time is. **Table 2** lists the relationship between the price Pr and recovery time Tr of the five SDP components assumed in this study.

Table 2. The relationship between price and recovery time.

Type	Price	Time of Recovery
1	1200	150
2	2000	120
3	2500	80
4	6500	40
5	12000	25

We define the node tolerance R_i^n of node i as a function of recovery time Tr,

$$R_i^n = e^{Tr_{min} - Tr_i} \tag{11}$$

In addition, we need to define and calculate the cost of deploying SDP components.

$$Pr = \sum_{i=1}^{M} Pr_i \tag{12}$$

2. Edge tolerance

In this study, SDP components are connected by wired network. In a wired network, the links between nodes may not be connected due to communication link failure. In addition,

SDN control information and data information are transmitted using the same network. Therefore, the higher the link load, the greater the possibility of control information delay arrival. The physical failure probability of the link is expressed in P_k^l, and the link tolerance R_k^l in the path is defined as follows:

$$R_k^l = \left(1 - P_k^l\right) \cdot Tanh\left(\frac{1}{\eta^k}\right) \tag{13}$$

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{14}$$

where, η^k is the bandwidth utilization of the *k*-th link. The greater the network traffic, the lower the link tolerance. Since both network traffic f^k and link bandwidth B_k are real numbers that are not less than zero, that is, η_k positive real numbers, the range of activation function $Tanh$ is between 0 and 1, which decreases as the independent variable increases and decreases. Therefore, we use this function to express the relationship between link tolerance and network traffic.

There can be multiple paths from the external network access node to the internal network access node. We assume that there are m paths between nodes i and j, one of the paths r has n_r sub links, and express the tolerance of the sub link k on this path as R_k^l. Thus, the path tolerance between node i and node j is defined as the product of the link tolerance of each hop, and the average link tolerance is defined as the average value of the link tolerance of all paths. For an SDP component node i, its edge tolerance is defined as the average tolerance of all paths that can be selected by the network flow with i as the external network or internal network access node.

$$R_{i,j}^{l,r} = \prod_{k=1}^{n_r} R_k^l \tag{15}$$

$$R_{i,j}^l = \frac{\sum_{r=1}^{m} R_{i,j}^{l,r}}{m} \tag{16}$$

$$R_i^l = \frac{\sum_{j=1}^{J} R_{i,j}^l}{J} \tag{17}$$

Where m is the number of paths between access point i and access point j, and J is the number of other ends of all network flows with SDP component i as the access end.

To sum up, the attack tolerance of SDP components should be combined with the tolerance of its nodes and the tolerance of connected links, which is defined as follows:

$$R_i = R_i^n + R_i^l \tag{18}$$

The average attack tolerance of all SDP components in the network is defined as:

$$R = \frac{\sum_{i \in V_g} R_i}{|V_g|} \tag{19}$$

3.3 Model Establishment

In the network, the occurrence of malicious attacks and the size of data flow load vary with the region and time. In the area where the SDP component fails or the data flow load is too

high due to the attacked network, the user authentication or legal access delay increases with the increase of data transmission delay. Therefore, when deploying SDP components, we must pay attention to the maximum average delay from SDP components in different regions to user network equipment. Specifically, when deploying SDP components, we must optimize the delay, link load, tolerance and deployment cost. To sum up, the optimization problem of deploying SDP components is shown as follows:

$$min\ z = \mu_1 t_{max} + \mu_2 \Delta\phi - \mu_3 R + \mu_4 Pr \tag{20}$$

$$s.t.: C1:\ \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \tag{21}$$

$$C2: t_{max} \leq T_{max} \tag{22}$$

$$C3: \Delta\phi \leq \phi_{max} \tag{23}$$

$$C4: i \in V_g, R_i \geq r_{min} \tag{24}$$

$$C5: \forall i \in [1, M], Pr_i \leq Pr_{max} \tag{25}$$

(21) indicates that the sum of the weights is equal to 1. Here, μ_1, μ_2, $\mu_3$and μ_4 represent the impact of delay, load difference, attack tolerance and deployment cost on optimization objectives respectively. (22) indicates that the maximum delay from the user equipment, that is, the initiating host, to the SDP component cannot exceed T_{max}. In addition, (23) indicates that the load difference between SDP components (expressed in the form of standard deviation of bandwidth utilization) cannot exceed ϕ_{max}. (24) indicates that the attack tolerance of each SDP component cannot be less than r_{min}. t_{max}^{i} represents the maximum delay of all SDP components from the external network access point to the SDP component of the internal network resource to be accessed, and R represents the average attack tolerance of all SDP components.

4. Algorithm and Parameter

In this study, we focus on the dynamic deployment of SDP components after the SDP components fail due to malicious attacks. Therefore, we must first traverse the candidate nodes, measure their delay, load balance, attack tolerance and other indicators, and then dynamically deploy the required number of SDP components according to the network status.

4.1 Introduction to DDPG

SDP component deployment is a Multiple Objective Combinatorial Optimization problem (MOCO) [33] and NP hard problem. Heuristic algorithm can be used to solve MOCO problem. However, these algorithms are easy to fall into local optimization. In addition, as the network state is constantly changing with the network load and malicious attacks, SDP components must be deployed according to the dynamic changes in the network. Therefore, it is very difficult to solve dynamic SDP component deployment using heuristic algorithm.

In contrast, reinforcement learning is more suitable for solving MOCO problems. DDPG algorithm [34] is a DRL algorithm that can output multidimensional actions. Due to the large number of nodes and complex states in the real network, SDP component deployment has a

huge state space and reconciliation space. While other DRL algorithms, such as Q-learning [35] with the increase of state space and action space, the size of q-table increases continuously, that is, the problem of dimension explosion occurs. Thus, the time required to find the feasible solution in the q-table is increasing, which will greatly affect the efficiency of the algorithm. DQN combines Q-learning with Convolutional Neural Networks (CNN) [36] to solve the problem of dimension explosion to a certain extent. However, DQN [37] has the problem of multi-dimensional output difficulty. DQN usually selects the action with the highest Q-value as the output, but it cannot output other actions with the highest Q-value, that is, the subsequent output is no longer the optimal solution. DDPG has the advantage of large state space and action space. Each deployment scheme of SDP component deployment is a multidimensional solution. A complete deployment solution can be output using DDPG algorithm. Therefore, each output is guaranteed to be optimal for the current state without high computational and storage costs.

To sum up, we use DDPG based algorithm to calculate SDP component deployment. In addition, we add different random numbers to each dimension of DDPG algorithm to improve the exploration ability of the algorithm. The corresponding state space, action space and reward are defined as follows.

4.2 Parameter setting

4.2.1 Space of State

The state space reflects the changes of the network state based on the deployment scheme at different times, and is an N×5 matrix . We define the state space of time t as follows:

$$S_t = \left(S_{q_1}^t, S_{q_2}^t, \ldots, S_{q_N}^t\right)^T = (D_t, T_t, R_t, \Phi_t, Pr_t) \tag{26}$$

where N is the number of nodes in the SDP network, and S_{q_i} is a column vector with a length of N, representing the current time state of the node q_i, including the direction and size of data outflow, delay, attack tolerance, load size, and deployment cost. The definition of network status S_{q_i} is as follows:

$$S_{q_i}^t = \left(d_{q_i}^t, t_{q_i}^t, R_{q_i}^t, \phi_{q_i}^t, Pr_{q_i}^t\right)^T \tag{27}$$

It is worth noting that when t=0, S_t indicates the initial state of the network. We define the initial state as that the user request flow starts to forward the data flow and plan the path according to the Dijkstra algorithm [38] and the shortest path principle.

In addition, D_t defines the data outflow direction and size of each node at time t. Its value represents the size of the data volume, and the position represents the outflow direction, that is, the flow to the node. Therefore, $D_t = (d_0^t, d_1^t, \ldots, d_N^t)^T$. T_t defines the delay of each node at time t, that is, the total delay sum of the access node is taken as this node, so $T_t = (t_0^t, t_1^t, \ldots, t_N^t)^T$. R_t defines the attack tolerance of each node at time t, hence $R_t = (R_0^t, R_1^t, \ldots, R_N^t)^T$. Φ_t defines the load size of each node at time t, that is, the sum of the loads of the link to which the SDP component is connected, therefore, $\Phi_t = (\phi_0^t, \phi_1^t, \ldots, \phi_N^t)^T$. Pr_t defines the cost of deploying SDP components at each node at time t, therefore, $Pr_t = (Pr_0^t, Pr_1^t, \ldots, Pr_N^t)^T$. All the above variables except D_t, when the node is not a deployment node, their values are set to 0, that is, other N dimensional column vectors can have at most M

non-zero values.

4.2.2 Space of Action

The action space represents the change strategy of network state at each step. In this study, it refers to the position vector of SDP components planed to be deployed. No more than M SDP components can be deployed in the SDP network. We use binary numbers to express the deployment of SDP components. The algorithm measures the priority of deployment location according to the reward, and uses 0/1 in the action space to indicate whether to deploy SDP components. There are at most M non-zero values. The action space used to deploy SDP components at time t is shown as follows:

$$A_t = (Z_1, Z_2, \ldots, Z_N)^T \tag{28}$$

4.2.3 Reward

The DDPG environment provides corresponding rewards according to the next network state formed by the action on the network, and guides the agent to constantly find a more optimal solution. Therefore, the reward should be designed based on (20) to achieve the optimization goal. However, $t_i, \Delta\phi, R, Pr$ have different dimensions and orders of magnitude, which affects the results of data analysis. To solve this problem, we use the Z-score function to standardize the index value that needs to calculate the reward:

$$f(x) = \frac{x-\overline{x}}{\sigma}, x \in X \tag{29}$$

After performing the action, the agent will get the reward according to the next network status.

$$r_t = f(z) \tag{30}$$

4.2.4 State Transition

The state transition is defined as (S_t, A_t, r_t, S_{t+1}), where S_t is the current network status, A_t is the action taken by the agent (the location and number of SDP components deployed), S_{t+1} is the new network status after getting reward r_t and perform action A_t.

Then, we design the algorithm according to the state space, action space and reward defined above. The initial deployment algorithm of SDP component based on DDPG is described in Algorithm 1. The redeployment algorithm to ensure the delay and other constraints are met when some SDP components are compromised under cyber attacks is described in Algorithm 2. In addition, we study the deployment of SDP components in the node varying situations, in which the deployment changes with the different cyber attacks.

4.3 Introduction to Algorithm

Next, according to the above optimization objectives and related parameters, we design the algorithm as follows. Our proposed model can be dynamically deployed for a given network topology and corresponding network state, as shown in Algorithm 1.

Algorithm 1: DDPG-based SDP components initial placement

Input: Request traffic from initial hosts RT, network topology G, weight factors μ_1, μ_2, μ_3

Output: Number of SDP components M, locations of SDP components L

1 Initialize the DDPG environment env with G, RT, μ_1, μ_2, μ_3;
2 **for** episode=0; episode<maxEpisode **do**
3 Divide network into NA areas by the LPA algorithm;
4 Randomly selecting one node in each areas as the initial placement locations IPL;
5 Initialize the initial network state S_0 according to the env and IPL;
6 **for** i=0; i<maxIteration **do**
7 Get the predicted next action A_i^p according to S_i by DDPG;
8 Add exploration noise N_t on the predicted next action A_i^p, then get the next action A_i;
9 Get the next state S_{i+1} and reward Re_i according to A_i by env;
10 Train the model in DDPG according to S_i, S_{i+1}, A_i and Re_i;
11 $S_i = S_{i+1}$;
12 Get the number of SDP components M and the locations of SDP components L according to the action with the maximum reward.

When an attacked SDP component fails in the network, the redeployment algorithm is triggered, as shown in Algorithm 2. Firstly, the network state, such as network traffic, network topology, weight factor, etc., is input into the algorithm as parameters and the DDPG environment variables are initialized. Then, randomly select an ordinary node around the failed SDP node as the initial redeployment position, start iterative search for the most suitable deployment position, and output it. In addition, there may be cases where the original normal SDP node accepts new service request traffic, and it is defined as a redirection node. Similar to Algorithm 1, in line 7, we introduce random variables N_t as exploration noise.

Algorithm 2: DDPG-based SDP components replacement when compromised

Input: Number of compromised SDP components CN, locations of compromised SDP components CL, network traffic flow F, network topology G, weight factors μ_1, μ_2, μ_3

Output: Locations of the new placed SDP components NL

1 Initialize the DDPG environment env with F, G, μ_1, μ_2, μ_3;
2 **for** episode=0; episode<maxEpisode **do**
3 Randomly selecting one node near each compromised SDP component as the initial replacement location IRL;
4 Initialize the initial network state according to the env and IRL;
5 **for** i=0; i<maxIteration **do**
6 Get the predicted next action A_i^p according to S_i by DDPG;
7 Add exploration noise N_t on the predicted next action A_i^p, then get the next action A_i;
8 Get the next state S_{i+1} and reward Re_i according to A_i by env;
9 Train the model in DDPG according to S_i, S_{i+1}, A_i and Re_i;
10 $S_i = S_{i+1}$;
11 Get the locations of the new placed SDP components NL according to the action with the maximum reward.

5. Evaluation

This section aims to evaluate the proposed solution by comparing it with different method, thus demonstrating the feasibility of deploying events in the WAN environment.

5.1 Platform

In order to evaluate the performance of the proposed solution, we set up a multi-node test platform based on Mininet 2.3.0 and RYU 4.34. We deployed VMware ESXI 6.5 as the management framework of virtualization service. The experimental platform is run on Ubuntu 22.04.3LTS operating system and has Intel Xeon(R) Gold 5118 CPU @ 2.30GHz with 48 cores and 48 GB RAM. In addition, DDPG-SDPCR algorithm is implemented based on Python 3.7.17 language and TensorFlow 1.13 framework.

5.2 Topology and Parameter

The proposed model and above mentioned methods are implemented over the topologies from Internet Topology Zoo [39]. Internet Topology Zoo currently includes 261 topologies, and the parameters of them are from real world. We selected 2 representative topologies, which often appear in the literature on network problem, to evaluate the proposed model and compare it to the baseline. Then the proposed model was run over them.

We use the real world network topologies to evaluate the proposed DDPG-SDPCR model and algorithms, that are the ATT North America network from the United States and the Geant network from Europe, which include 25 nodes and 57 edges, and 34 nodes and 52 nodes respectively. The distance and relative position between nodes are calculated by Haversine formula [40] according to the latitude and longitude coordinates provided by Internet Topology Zoo, and the shortest path between nodes is calculated by Dijkstra algorithm [38]. The propagation delay between nodes and transmission delay of wired network are calculated by (2) and (3). The parameters in the experiment are listed in **Table 3** and **Table 4**.

Table 3. Parameters of AAT North America.

Name or description	Value
Number of normal switches	15
Number of SDP components	10
μ_1	0.4
μ_2	0.3
μ_3	0.2
μ_4	0.1

Table 4. Parameters of Geant.

Name or description	Value
Number of normal switches	20
Number of SDP components	14
μ_1	0.4
μ_2	0.3
μ_3	0.2
μ_4	0.1

In order to analyze and explore the influence of SDP nodes damaged by malicious attacks on the flow delay of the whole SDP network, we tested the network delay in 4 network environments with different traffic sizes: 1Mbits, 2Mbits, 4Mbits and 8Mbits. In order to analyze and explore the difference of the impact of malicious attacks on the flow delay of the

whole network, we set the failure rate of SDP nodes to 0.1, and randomly selected two kinds of three SDP nodes to conduct experimental tests. The input parameters in DDPG-SDPCR are shown in **Table 3** and **Table 4**, and the details of the two faults are shown in **Table 5**. We firstly determine the deployment position of SDP nodes in the input topology according to Algorithm 1. Then, due to cyber attacks, according to the failure probability, some SDP nodes fail. Corresponding topology changes are shown in **Figs. 4-5** and **Figs. 8-9**.

Table 5. The specific situation of network failure.

Topology	ATT North America	Geant
Compromised SDP Node Under Attack 1	4, 10, 22	2, 10, 12
Compromised SDP Node Under Attack 2	7, 16, 25	3, 9, 11

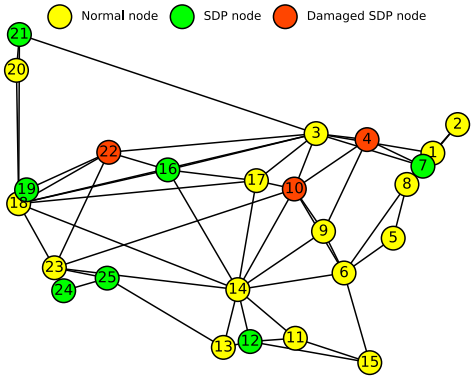

Fig. 4. Compromised SDP nodes under attack 1 in ATT North America.

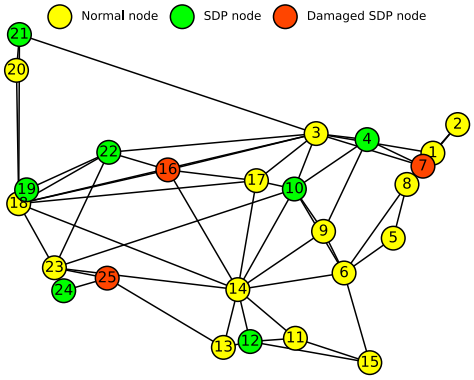

Fig. 5. Compromised SDP nodes under attack 2 in ATT North America.

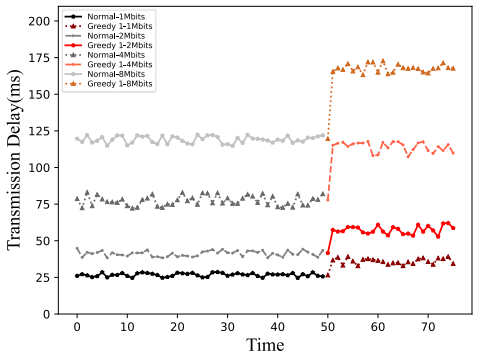

Fig. 6. The TD of ATT North America changes with different situations under attack 1.(Greedy)

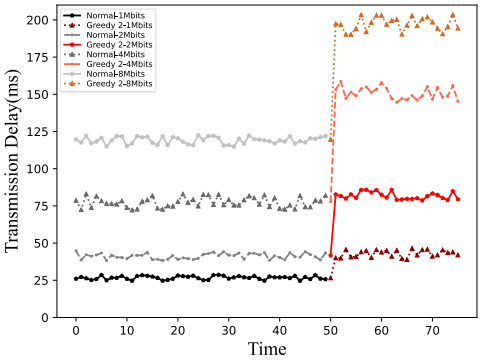

Fig. 7. The TD of ATT North America changes with different situations under attack 2.(Greedy)

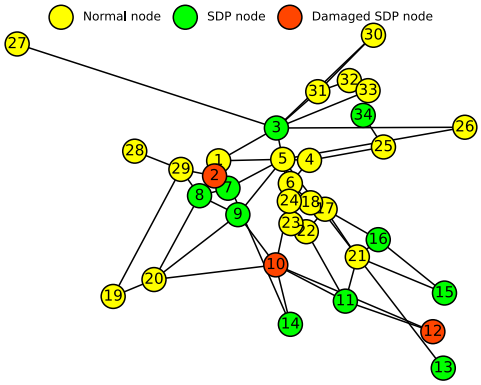

Fig. 8. Compromised SDP nodes under attack 1 in Geant.

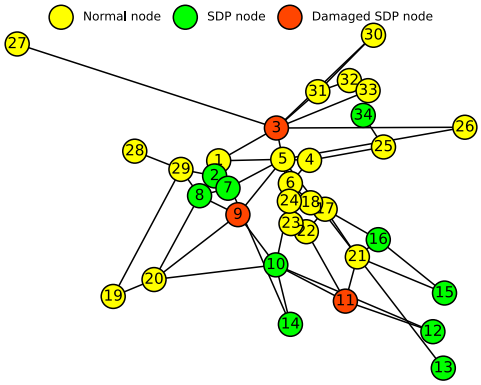

Fig. 9. Compromised SDP nodes under attack 2 in Geant.

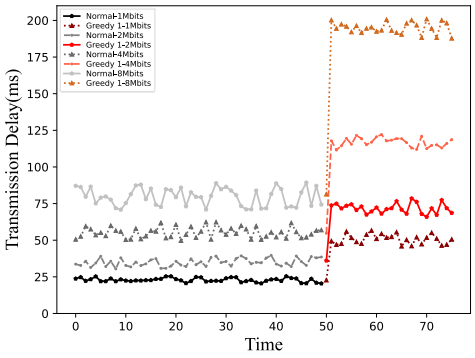

Fig. 10. The TD of Geant changes with different situations under attack 1.(Greedy)

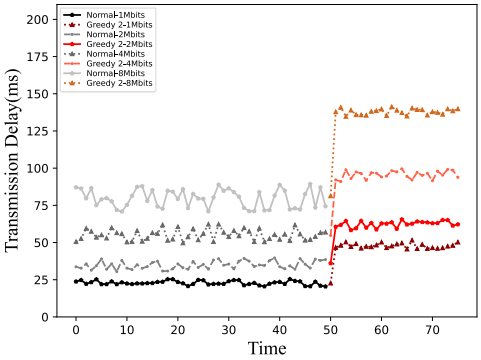

Fig. 11. The TD of Geant changes with different situations under attack 2.(Greedy)

According to the failure probability, we randomly select 2 different failure situations under attacks in the ATT North America network, as shown in **Figs. 4-5**. The yellow nodes are the normal nodes, the lime nodes are the SDP nodes, and the orangered nodes are the damaged SDP nodes. Node 4, 10 and 22 in **Fig. 4**, and Node 7, 16 and 25 in **Fig. 5**, which are affected by Attack 1 and Attack 2. Similarly, **Figs. 8-9** show the situation when SDP nodes in Geant network are attacked and fail, in which the failed nodes affected by Attack 1 are 2, 10 and 12, and those affected by Attack 2 are 3, 9, 11. Then, according to the serial number of the failed nodes and the corresponding user requests relationships, we use greedy algorithm as baseline to shunt the load of the failed SDP nodes, and compare it with our proposed DDPG-SDPCR algorithm.

As shown in **Figs. 6-7**, under two kinds of attacks, the average Transmission Delay (TD) of ATT North America network increased by 9.257ms (34.82%), 15.630ms (37.54%), 35.444ms (45.56%), and 48.824ms (40.81%), and 16.437ms (61.83%), 40.739ms (97.86%), 74.121ms (95.27%), and 77.293ms (64.61%) when the traffic size was 1Mbits, 2Mbits, 4Mbits, and 8Mbits respectively. Similarly, as shown in **Figs. 10-11**, under two kinds of attacks, the average TD of Geant network increased by 29.922ms (131.70%), 36.776ms (101.97%), 61.176ms (111.81%), and 114.670ms (140.96%), and 5.714ms (113.18%), 26.491ms (73.45%), 40.02ms (73.15%), and 56.31ms(69.23%), respectively. It can be found that in both topologies, even if the network load redirection strategy based on greedy algorithm is adopted, the network attack will still have a significant impact on the performance of SDP network after

the SDP node fails. In addition, due to the difference of node location and the size of traffic requested by the host, the SDP node failure caused by different attacks has significant differences in the increase of TD. To sum up, a more effective fault tolerance strategy is necessary.

In order to evaluate the mitigation effect of our DDPG-SDPCR mechanism on the increase of network delay caused by SDP node failure and damaged in the above two networks, we conducted relevant tests. We tested each network for 12 times, including 2 kinds of attacks and 4 kinds of traffic size (1Mbits, 2Mbits, 4Mbits and 8Mbits), and discussed the normal situation (not attacked), redirection recovery based on greedy algorithm (as a comparison baseline) and redeployment recovery based on DDPG-SDPCR mechanism, as shown in **Figs. 12-23**.

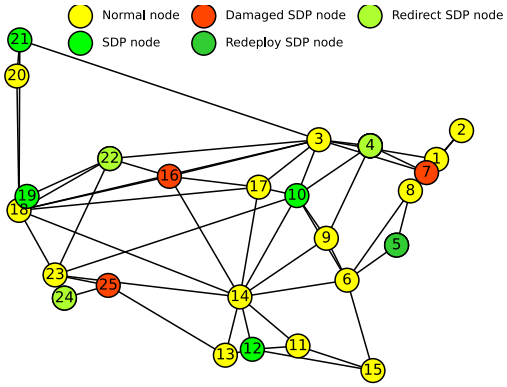

Fig. 12. ATT North America redeployed by DDPG-SDPCR under attack 1.

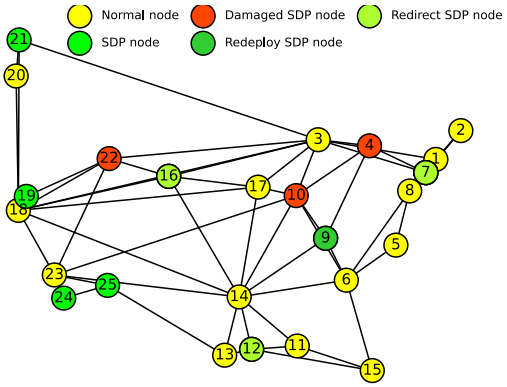

Fig. 13. ATT North America redeployed by DDPG-SDPCR under attack 2.

Figs. 12-13 are the redeployment topology diagrams obtained by Algorithm 2 after SDP node failure caused by two kinds of attacks on ATT North America network. Among them, except normal nodes, SDP nodes and damaged nodes, limegreen nodes are redeployed SDP nodes and yellowgreen nodes are redirected SDP nodes. As shown in **Fig. 12**, in order to cope with SDP nodes failures caused by Attack 1, Algorithm 2 gives a redeployment node Node 5, and 3 redirection nodes Node 4, 22 and 24. As a normal SDP node, the latter bears a part of the requested traffic of the originating host because there are faulty SDP nodes around. In this way, the number of redeployment nodes can be reduced and the redeployment cost can be reduced. Similarly, **Fig. 13** shows the redeployment in case of Attack 2. The redeployment node is Node 9, and the redirection nodes are Node 7, 12 and 16.

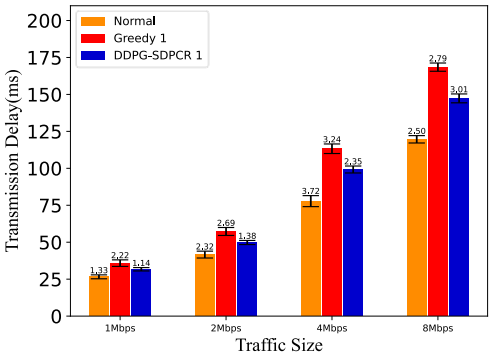

Fig. 14. The TD comparison of ATT North America among different situations under Attack 1.

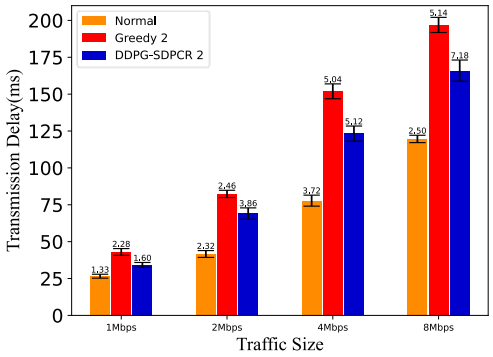

Fig. 15. The TD comparison of ATT North America among different situations under Attack 2.

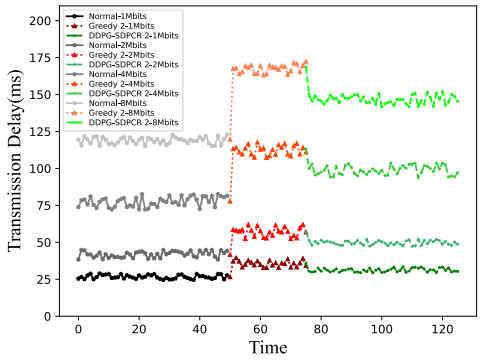

Fig. 16. The TD of ATT North America changes with different situations under Attack 1.(DDPG-SDPCR)

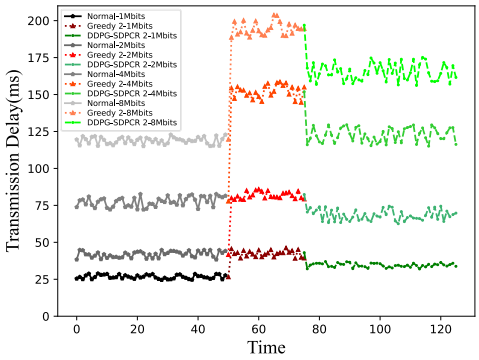

Fig. 17. The TD of ATT North America changes with different situations under Attack 1.(DDPG-SDPCR)

As shown in **Fig. 14**, under normal circumstances, TD of ATT North America network are 26.582ms, 41.631ms, 77.799ms, and 119.631ms when the traffic size are 1Mbits,2Mbits,4Mbits, and 8Mbits, respectively. In the case of Attack 1, after redirection by greedy algorithm, the corresponding TD are 35.839ms, 57.261ms, 113.243ms, and 168.455ms, respectively. After the redeployment of DDPG-SDPCR mechanism, the corresponding TD are 31.698ms, 49.806ms, 99.202ms, and 147.320ms. Combined with the above data, after the redeployment of DDPG-SDPCR, compared with the normal situation, TD increased by 5.116ms (19.24%), 8.175ms (19.64%), 21.404ms (27.51%), and 27.689ms (23.14%), respectively, and compared with greedy algorithm, decreased by 4.142ms (11.56%), 7.455ms (13.02%), 14.041ms (12.40%), 21.136ms (12.55%), respectively. As shown in **Fig. 15**, in the case of Attack 2, after redirection by greedy algorithm, TD under different traffic size are 43.019ms, 82.37ms, 151.920ms, and 196.924ms, respectively. After the redeployment of DDPG-SDPCR mechanism, the corresponding TD are 34.281ms, 69.033ms, 123.259ms, and 165.954ms, respectively. Combined with the above data, after the redeployment of DDPG-SDPCR, compared with the normal situation, TD increased by 7.699ms (28.96%), 27.402ms (65.82%), 45.461ms (58.43%), and 46.323ms (38.72%), respectively, and compared with greedy algorithm, decreased by 8.738ms (20.31%), 13.337ms (16.19%), 28.660ms (18.87%), and 30.971ms (15.73%), respectively. With the increase of traffic, the TD of the network will

increase. It can be found that compared with greedy algorithm, our proposed DDPG-SDPCR mechanism can alleviate the TD increase caused by attacks to a great extent, and significantly reduce the delay increase caused by greedy algorithm. **Figs. 16-17** are line charts of TD changing with time under different traffic sizes in ATT North America network under Attack 1 and Attack 2, respectively. It can be found that the cyber attack occurred at 50s, then the greedy algorithm started to run, and TD increase greatly. At about 75s, DDPG-SDPCR mechanism started to run, and TD dropped significantly. Through the above two line charts, the advantages of DDPG-SDPCR compared with greedy algorithm can be intuitively displayed. In addition, two different attacks have different effects, and Attack 2 makes the TD of the network increase more.

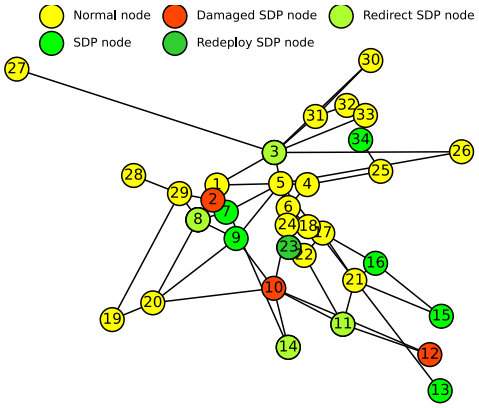

Fig. 18. Geant redeployed by DDPG-SDPCR under Attack 1.

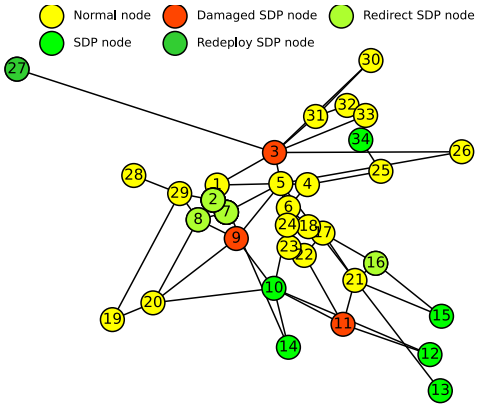

Fig. 19. Geant redeployed by DDPG-SDPCR under Attack 2.

Figs. 18-19 show the redeployment topology diagram of Geant network after SDP node failure caused by two kinds of attacks, which is calculated by algorithm 2. As shown in **Fig. 18**, in order to deal with some SDP node failures caused by Attack, algorithm 2 gives redeployment node Node 23 and redirection nodes Node 3, 8, and 11. Similarly, **Fig. 19** shows the redeployment in the case of Attack 2. The redeployment node is Node 27, and the redirection nodes are Node 2, 7, 8, and 16.

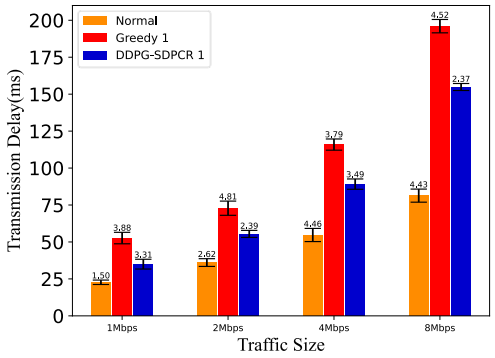

Fig. 20. The TD comparison of Geant among different situations under Attack 1.

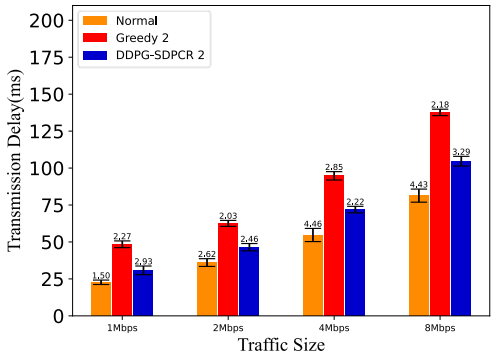

Fig. 21. The TD comparison of Geant among different situations under Attack 2.

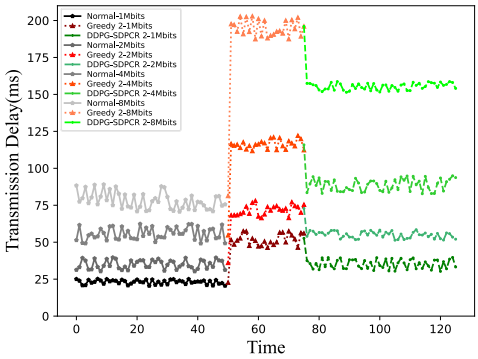

Fig. 22. The TD of Geant changes with different situations under Attack 1.(DDPG-SDPCR)

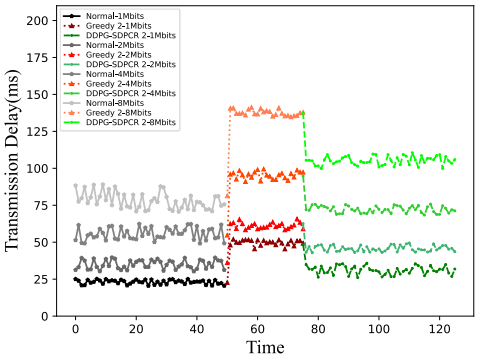

Fig. 23. The TD of Geant changes with different situations under Attack 2.(DDPG-SDPCR)

As shown in **Fig. 20**, under normal circumstances, the TD of Geant network are 22.720ms, 36.065ms, 54.712ms, and 81.347ms when the traffic size are 1Mbits,2Mbits,4Mbits,and 8Mbits, respectively. In the case of Attack 1, after redirection by greedy algorithm, the corresponding TD are 52.641ms, 72.841ms, 115.888ms, and 196.017ms, respectively. After the redeployment of DDPG-SDPCR mechanism, the corresponding TD are 34.989ms, 55.455ms, 89.137ms, and 154.853ms, respectively. Combined with the above data, after the redeployment, TD increased by 12.269ms (54.00%), 19.390ms (53.76%), 34.425ms (62.92%), and 73.506ms (90.36%), respectively, compared with the normal situation, and the corresponding TD decreased by 17.652ms (33.53%), 17.386ms (23.87%), 26.751ms (23.08%), and 41.164ms (21.00%), respectively, compared with the greedy algorithm. As shown in **Fig. 21**, in the case of Attack 2, after redirection by greedy algorithm, the corresponding TD under different traffic size are 48.433ms, 62.557ms, 94.736ms, and 137.660ms, respectively. After the redeployment of DDPG-SDPCR mechanism, the corresponding TD are 30.794ms, 46.481ms, 71.890ms, and 104.647ms, respectively. Combined with the above data, after the redeployment, TD increased by 8.074ms (35.54%), 10.416ms (28.88%), 17.178ms (31.40%), and 23.299ms (28.64%), respectively, compared with the normal situation, and the corresponding TD decreased by 17.640ms (36.42%), 16.076ms (25.70%), 22.846ms (24.115%), and 33.014ms (23.98%), respectively, compared with the greedy algorithm. With the increase of traffic, the TD of the network will increase. It can be found that compared with the greedy algorithm, our proposed DDPG-SDPCR mechanism can greatly alleviate the TD increase caused by the attack and significantly reduce the delay increase caused by the greedy algorithm. **Figs. 22-23** are line charts of TD changing with time under different traffic sizes in Geant network under Attack 1 and Attack 2, respectively. It can be found that the cyber attack occurred at 50s, then the greedy algorithm started to run, and TD increase greatly. At about 75s, DDPG-SDPCR mechanism started to run, and TD dropped significantly. Through the above two line charts, the advantages of DDPG-SDPCR compared with greedy algorithm can be intuitively displayed. In addition, two different attacks have different effects, and Attack 1 makes the TD of the network increase more.

6. Conclusion

In this study, a wide area SDP framework and corresponding attack tolerance mechanism are proposed, which can provide users with wide area remote intranet resource security access

services. In addition, considering the potential cyber attacks, the SDP components can be redeployed after some SDP components fail due to malicious attacks, and the delay and bandwidth requirements of users can be met. Based on DDPG-SDPCR redeployment mechanism, considering the constraints of network traffic, link bandwidth and delay, the appreciate nodes in the network are selected as the deployment locations of the newly added SDP components. In addition, this study also carries out experimental tests and results analysis for different network topologies, different attack scenarios and different traffic sizes, and compares them with the baseline. The results show that our proposed framework based on DDPG can effectively reduce the impact of malicious attacks on the SDP wide area network, and it is superior to the baseline in terms of delay, link load and attack tolerance.

Acknowledgement

This work was supported by Science and Technology Innovation leading Talents Subsidy Project of Central Plains (Grant No. 244200510038).

References

[1] Moubayed, A., Refaey, A., Shami, A., "Software-Defined Perimeter (SDP): State of the Art Secure Solution for Modern Networks," *IEEE Network*, vol.33, no.5, pp.226-233, 2019. Article(CrossRefLink)

[2] Palmo, Y., Tanimoto, S., Sato, H. et al., "A Consideration of Scalability for Software Defined Perimeter Based on the Zero-trust Model," in *Proc. of 2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp.717-724, 2021. Article(CrossRefLink)

[3] Phu, A. T., Li, B., Ullah, F. et al., "Defending SDN against packet injection attacks using deep learning," *Computer Networks*, vol.234, 2023. Article(CrossRefLink)

[4] Singh, J., Refaey, A., Koilpillai, J., "Adoption of the Software-Defined Perimeter (SDP) Architecture for Infrastructure as a Service," *Canadian Journal of Electrical and Computer Engineering*, vol.43, no.4, pp.357-363, 2020. Article(CrossRefLink)

[5] Sallam, A., Refaey, A., Shami, A., "On the Security of SDN: A Completed Secure and Scalable Framework Using the Software-Defined Perimeter," *IEEE Access*, vol.7, pp.146577-146587, 2019. Article(CrossRefLink)

[6] Bakhshi, B., Mangues-Bafalluy, J., Baranda, J., "Multi-provider NFV network service delegation via average reward reinforcement learning," *Computer Networks*, vol.224, 2023. Article(CrossRefLink)

[7] Fang, J., Zhao, G., Xu, H. et al., "Reveal: Robustness-aware VNF placement and request scheduling in edge clouds," *Computer Networks*, vol.233, 2023. Article(CrossRefLink)

[8] Wu, Z., Peng, S., Liu, L. et al., "Detection of Improved Collusive Interest Flooding Attacks Using BO-GBM Fusion Algorithm in NDN," *IEEE Transactions on Network Science and Engineering*, vol.10, no.1, pp.239-252, 2023. Article(CrossRefLink)

[9] Kalafatidis, S., Skaperas, S., Demiroglou, V. et al., "Logically-Centralized SDN-Based NDN Strategies for Wireless Mesh Smart-City Networks," *Future Internet*, vol.15, no.1, 2022. Article(CrossRefLink)

[10] Rodrigues, D. O., Braun, T., Maia, G. et al., "Mobility-aware Latency-constrained Data Placement in SDN-enabled Edge Networks," in *Proc. of NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp.1-9, 2023. Article(CrossRefLink)

[11] Naseri, A., Ahmadi, M., PourKarimi, L., "Placement of SDN controllers based on network setup cost and latency of control packets," *Computer Communications*, vol.208, pp.15-28, 2023. Article(CrossRefLink)

[12] Peng, S., Niu, D., Jin, K. et al., "Toward low-delay and low-cost controller placement in SDNs: a hybrid heuristic method," in *Proc. of the 2023 9th International Conference on Computing and Artificial Intelligence*, pp.743-751, 2023. Article(CrossRefLink)
[13] Yue, G., Wang, Y., Liu, Y., "Rule Placement and Switch Migration-based Scheme for Controller Load Balancing in SDN," in *Proc. of 2022 IEEE Symposium on Computers and Communications (ISCC)*, pp.1-6, 2022. Article(CrossRefLink)
[14] Xiao, J., Pan, X., Liu, J. et al., "Load balancing strategy for SDN multi-controller clusters based on load prediction," *The Journal of Supercomputing*, vol.80, pp.5136-5162, 2024. Article(CrossRefLink)
[15] Ramya, G., Manoharan, R., "Traffic-aware dynamic controller placement in SDN using NFV," *The Journal of Supercomputing*, vol.79, no.2, pp.2082-2107, 2023. Article(CrossRefLink)
[16] Maity, I., Dhiman, R., Misra, S., "EnPlace: Energy-Aware Network Partitioning for Controller Placement in SDN," *IEEE Transactions on Green Communications and Networking*, vol.7, no.1, pp.183-193, 2023. Article(CrossRefLink)
[17] Kondo, T., Guillen, L., Izumi, S. et al., "An Energy Efficient SDN Controller Placement with Delay Constraints," in *Proc. of 2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp.119-124, 2023. Article(CrossRefLink)
[18] Xu, C., Xu, C., Li, B. et al., "Load-aware dynamic controller placement based on deep reinforcement learning in SDN-enabled mobile cloud-edge computing networks," *Computer Networks*, vol.234, 2023. Article(CrossRefLink)
[19] Heller, B., Sherwood, R., McKeown, N., "The controller placement problem," in *Proc. of HotSDN '12: Proceedings of the first workshop on Hot topics in software defined networks*, pp.7-12, 2012. Article(CrossRefLink)
[20] DOcplex Python Modeling API, ILOG CPLEX Optimization Studio. [Online]. Available: https://www.ibm.com/docs/zh/icos/22.1.1?topic=docplex-python-modeling-api
[21] Cui, J., Zhang, J., He, J. et al., "DDoS detection and defense mechanism for SDN controllers with K-Means," in *Proc. of 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pp.394-401, 2020. Article(CrossRefLink)
[22] Chakrabarty, D., Goyal, P., Krishnaswamy, R., "The Non-Uniform k-Center Problem," *ACM Transactions on Algorithms (TALG)*, vol.16, no.4, pp.1-19, 2020. Article(CrossRefLink)
[23] Drezner, Z., Facility Location: A Survey of Applications and Methods, New York, NY, USA: Springer-Verlag, 1995. Article(CrossRefLink)
[24] Owen, S. H., Daskin, M. S., "Strategic facility location: A review," *European Journal of Operational Research*, vol.111, no.3, pp.423-447, 1998. Article(CrossRefLink)
[25] Jacob, J., Shinde, S., and Narayan, D. G., "An Efficient Controller Placement Algorithm using Clustering in Software Defined Networks," *Journal of Telecommunications and Information Technology*, vol.4, no.4, pp.9-17, 2023. Article(CrossRefLink)
[26] Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Hock, D., Jarschel, M., and Hoffmann, M., "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, vol.12, no.1, pp.4-17, 2015. Article(CrossRefLink)
[27] Singh, A. K., Maurya, S., Kumar, N., and Srivastava, S., "Heuristic approaches for the reliable SDN controller placement problem," *Transactions on Emerging Telecommunications Technologies*, vol.31, no.2, 2020. Article(CrossRefLink)
[28] Singh, A. K., Srivastava, S., and Banerjea, S., "Evaluating heuristic techniques as a solution of controller placement problem in SDN," *Journal of Ambient Intelligence and Humanized Computing*, vol.14, no.9, pp.11729-11746, 2023. Article(CrossRefLink)
[29] Gholamrezaei, R., Mirjalily, G., Emadi, S., "Learning-based multi-constraint resilient controller placement and assignment in software-defined networks using covering graph," *Transactions on Emerging Telecommunications Technologies*, vol.34, no.4, 2023. Article(CrossRefLink)
[30] Llorens-Carrodeguas, A., Cervelló-Pastor, C., Valera, F., "DQN-based intelligent controller for multiple edge domains," *Journal of Network and Computer Applications*, vol.218, 2023. Article(CrossRefLink)

[31] Li, B., Deng, X., Chen, X. et al., "MEC-Based Dynamic Controller Placement in SD-IoV: A Deep Reinforcement Learning Approach," *IEEE Transactions on Vehicular Technology*, vol.71, no.9, pp.10044-10058, 2022. Article(CrossRefLink)
[32] Gong, Y., Wei, Y., Yu, F. R. et al., "Slicing-based resource optimization in multi-access edge network using ensemble learning aided DDPG algorithm," *Journal of Communications and Networks*, vol.25, no.1, pp.1-14, 2023. Article(CrossRefLink)
[33] Barrett, T., Clements, W., Foerster, J., Lvovsky, A., "Exploratory Combinatorial Optimization with Reinforcement Learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol.34, no.4, pp.3243-3250, Apr. 2020. Article(CrossRefLink)
[34] He, H., Zhou, F., Zhao, Y. et al., "Hypergraph convolution mix DDPG for multi-aerial base station deployment," *Journal of Cloud Computing*, vol.12, 2023. Article(CrossRefLink)
[35] Xu, Q., Su, Z., Lu, R., "Game Theory and Reinforcement Learning Based Secure Edge Caching in Mobile Social Networks," *IEEE Transactions on Information Forensics and Security*, vol.15, pp.3415-3429, 2020. Article(CrossRefLink)
[36] Priyadarshini, I., Mohanty, P., Alkhayyat, A. et al., "SDN and application layer DDoS attacks detection in IoT devices by attention-based Bi-LSTM-CNN," *Transactions on Emerging Telecommunications Technologies*, vol.34, no.11, 2023. Article(CrossRefLink)
[37] Du, T., Li, C., Luo, Y., "Latency-aware computation offloading and DQN-based resource allocation approaches in SDN-enabled MEC," *Ad Hoc Networks*, vol.135, 2022. Article(CrossRefLink)
[38] Dijkstra, E. W., A Note on Two Problems in Connexion with Graphs, Edsger Wybe Dijkstra: His Life, Work, and Legacy, pp.287-290, 2022. Article(CrossRefLink)
[39] Knight, S., Nguyen, H. X., Falkner, N. et al., "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol.29, no.9, pp.1765-1775, 2011. Article(CrossRefLink)
[40] Robusto, C. C., "The Cosine-Haversine Formula," *The American Mathematical Monthly*, vol.64, no.1, pp.38-40, 1957. Article(CrossRefLink)

Zheng Zhang(first author), received the bachelor's degree from PLA Strategic Support Force Information Engineering University in 2019. He is currently pursuing Doctor Degree in PLA Strategic Support Force Information Engineering University, Zhengzhou, China. His research interests include software-defined networking, network security, and network infrastructure security.

Quan Ren, is currently a research associate at the National Digital Switching System Engineering and Technological Research and Development Center. He received the B.S. degrees in information engineering from the Southeast University, Nanjing, China, in 2016. He received the Ph.D. degree. in cyberspace security from the PLA Information Engineering University, Zhengzhou, China, in 2022. His research interests include cyberspace security and network architecture.

Jie Lu, received the Ph.D. degree. from the PLA Strategic Support Force Information Engineering University in 2022. His research interests include software-defined networking, network management, and network security.

Yuxiang Hu(corresponding author), professor, deputy director of the third research office of Information Technology Institute of PLA Strategic Support Force Information Engineering University Zhengzhou, China. His research areas are next generation network architecture and network security.

Hongchang Chen, professor, deputy director of the National Digital Switching System Engineering Technology Research Center, and leader of the innovation team of the National Science and Technology Progress Award for Network Communication and Switching Technology. The main research areas are cyberspace security, big data and artificial intelligence.