IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 10, 2019, accepted May 8, 2019, date of publication May 15, 2019, date of current version June 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917027

Multi-Controller Deployment Algorithm in Hierarchical Architecture for SDWAN

HOU XIAOLAN[1], WU MUQING[1], LV BO[2], AND LIU YIFENG[2]
[1]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]China Academy of Electronics and Information Technology, Beijing 100041, China

Corresponding author: Hou Xiaolan (houxiaolan@bupt.edu.cn)

This work was supported in part by the 111 Project under Grant b17007, in part by the Director Funds of the Beijing Key Laboratory of Network System Architecture and Convergence under Grant 2017BKL-NSAC-ZJ-01, and in part by the National Natural Science Foundation of China (NSFC) under Grant 61872401.

ABSTRACT A software-defined network (SDN) is a new paradigm that separates control and forwarding. With the growth of SDN deployment scale, the centralized control plane becomes a bottleneck restricting the expansion of SDN networks. To avoid the problem of limited processing power in the wide area network (WAN), we focus on the multi-controller deployment in hierarchical architecture for software-defined WAN (SDWAN). The hierarchical architecture divides the control plane into multiple levels. The root controller has a global view. The domain controller is only responsible for the control in local networks. In this paper, we use the improved Louvain algorithm to discover the hierarchical community structure. Considering the load balancing and request delay in each partition and reliability of the control plane, we formulate the modeling and propose a hierarchical multi-controller deployment algorithm for the scalability of the SDWAN. The simulation results show that compared with the traditional algorithms, the proposed algorithm not only reduces the average request delay to a certain extent and effectively improves the performance of load balancing in each partition, but also efficiently enhances the reliability of the control plane in the SDWAN and achieves the good effect on the execution efficiency.

INDEX TERMS Software-defined WAN, controller deployment, hierarchical architecture, Louvain algorithm.

I. INTRODUCTION

Software-defined network (SDN) [1] separates the control plane from the forwarding plane, brings the network programmability, and simplifies network configuration. The network managers can automatically, quickly, and dynamically configure and optimize network resources with the cooperation of control layer applications [2]. SDN has the characteristics of centralized control and network programmability, which enhances the flexibility and openness of the network. SDN has become an excellent solution in the field of new network technology. In view of these advantages of SDN, the SDN architecture is introduced into the wide area network (WAN), that is, the software-defined WAN (SDWAN). With the expansion of SDWAN applications, the problems caused by centralized control architecture are also emerging. The resources of the control plane are generally limited. As the scale of the network increases, the number of network events also increases. When the controller plane is not enough to handle these network events, the scale of the network will be limited. In the large network control scenario, the scalability of SDWAN architecture is also put forward higher requirements. Therefore, it is necessary to conduct an in-depth study on the scalability issues [3], [4].

At present, there are two main solutions to solve the scalability of control plane in the SDN. One is to improve the performance of control plane. The processing capability of control plane is enhanced by improving the performance of single controller. On the other hand, multi-controller architecture is adopted. By deploying multiple controllers to share processing requests from the network, the load on a single controller can be reduced and the processing power of the entire control plane can be improved. In fact, the processing power of a single controller is limited, and a single controller cannot meet the requirements of processing capability in the increasingly large network. Therefore, the scalability

The associate editor coordinating the review of this manuscript and approving it for publication was Tariq Umer.

problem can only be alleviated to some extent by improving the performance of single processor. In order to meet the needs of large-scale network, it is necessary to design a multi-controller architecture as the control plane [5], [6].

At present, there are two multi-controller architectures: flat architecture and hierarchical architecture. In the flat architecture, all controllers in the control plane are at the same level and have equal authority to make network decisions, such as ONIX [7] and ONOS [8]. However, since all controllers in the control plane need to maintain the same global network view through consistency synchronization, it takes a lot of computing power for data synchronization, especially when the number of controllers increases. Therefore, the flat architecture has limited scalability due to the limitation of the data plane synchronization efficiency between the controllers. The hierarchical architecture maintains scalability and flexible programmability of the SDN control plane. It divides the controllers into the root controller and the domain controllers by function in a hierarchical way, which reduces the complexity of global information synchronization and improves the communication efficiency. The management area of the data plane network is divided into domains, which improves the scalability of SDN network. The flat architecture needs to maintain a common network view. In large networks, due to the remote distance between controllers, the cost of global information synchronization is high, which consumes a lot of time and network resources. But the processing logic of controllers is relatively simple and easy to implement. The controllers in the hierarchical architecture only need to master its own network view. So it does not require strong consistency of the network views of controllers, only need to synchronize specific information to specific roles. The amount of information synchronized between the controllers can be effectively reduced. But it needs to design and implement different functions for different network roles, which is more complex to implement. If the load on the root controller increases, it may cause the root controller to crash, creating a network bottleneck. Therefore, the scalability of the root controller is also to be studied. In this paper, we focus on network partitioning and the placement of root controller and domain controller.

The hierarchical architecture divides the control plane of the SDWAN into multiple levels. The super controller, namely root controller, has a global view and is responsible for making global decisions for SDWAN. The domain controller is only responsible for the control of local networks, which can meet the management and control needs in SDWAN. The two-layer hierarchical architecture is shown in Fig.1.

The communication between the root controller and the domain controller is accomplished through the communication protocol OXP. The data plane is composed of SDN switches and communicates with the domain controller through the OpenFlow protocol. The intra-domain requests are not uploaded to the root controller and can only be processed by the domain controller. When there is an inter-domain request, the domain controller sends the request to the root controller. Then the root controller processes the request according to the information of the entire network, and notifies the relevant domain controller of the processing result. After receiving the response of the root controller to the forwarding request, the domain controller translates it into a southbound protocol message and sends it to all switches. In this way, through the coordination of root controller, the cooperative work among multi-controller is realized.

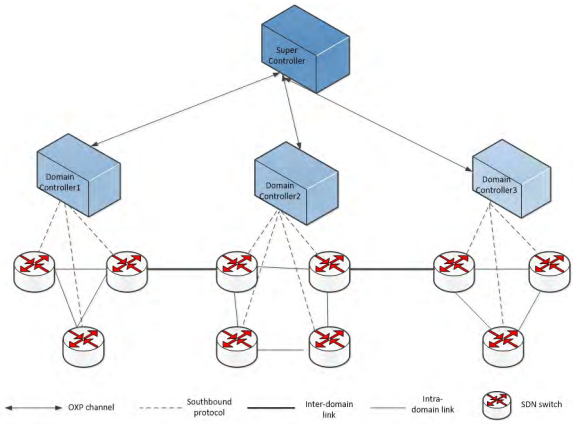

FIGURE 1. The hierarchical control architecture.

With the deepening of SDN networks practice, research has found that controller deployment affects the ability of controller to handle network events [9]. An excellent controller deployment case can balance the load between controllers and effectively reduce data loss in the control plane. An effective controller deployment method can efficiently distribute management functions between controllers and improve the scalability of management. The controllers placement problem(CPP) has become one of the hot issues in the SDN research. The research of CPP mainly concentrates on several aspects: propagation delay [9], reliability [10], load [11], overhead [12], and multi-objective [13], [14]. In the hierarchical architecture, for inter-domain requests, the communication performance between switches and domain controllers, as well as domain controllers and root controller, need be considered. Therefore, the partitioning and controllers placement in flat architecture can not meet the requirements of hierarchical architecture. However, most of the current research on CPP is focused on the flat architecture. The problem of controller deployment in hierarchical control architecture is rarely studied. In this paper, we study the controller deployment based on the hierarchical architecture for the scalability of SDWAN. The Louvain algorithm [15] is a community discovery algorithm based on modularity, which performs well in efficiency and effectiveness and does not require to specify the number of partitions. The algorithm can discover hierarchical community structure, which is exactly in line with the characteristic of hierarchical architecture. We make full use of this feature to solve the controller deployment problem in hierarchical control architecture. We propose an improved Louvain algorithm to partition the data plane to

reduce the request delay between the switch and the domain controller and optimize the load balancing of each domain. Our contributions are summarized as follows:

First of all, our work takes into account the request delay between the switch and the controller, the load balancing in each domain, and the reliability of the control plane, respectively formulating mathematical model.

Then, we propose the improved Louvain algorithm to partition the data plane and deploy the domain controller and root controller for reliability.

The simulation results show that the proposed algorithm can reduce the request delay to a certain extent, make the load of each partition more balanced, and improve the reliability of the control plane.

The structure of the paper is as follows. In Section 2, we introduce related work. In Section 3, we describe our proposed algorithm. In Section 4, we present the performance evaluation. In Section 5, we conclude the paper.

II. RELATED WORK

The SDN controller masters the global resource view of the SDWAN in a centralized manner. The controller deployment scheme in SDWAN directly affects the ability of controller to handle network events. The controller deployment problem is a current research hotspot. There has been a lot of research on the controller deployment in SDN. Heller *et al.* [9] first propose the controller deployment problem, put forward two indicators to measure the average and maximum delay in the network, and analyze the impact of these two indicators on controller deployment in real network environment. The current control plane in SDN mostly uses a multi-controller architecture to improve the scalability of the control plane. In the multi-controller control plane scenario, how to reasonably deploy the multi-controller is the main research direction.

The authors in [11] present a new community detection controller deployment method (CDCD). To restrict the number of nodes in each community and balance the differences in the number of nodes between different communities, Louvain Heuristic Algorithm (LHA) is improved and the scale constraint factor is proposed. The authors in [14] propose an adaptive evolutionary algorithm with greedy heuristic to produce high quality initial populations, intelligent mechanisms to promote diversification and intensification, and new fast Pareto finder for large-scale multi-objective controller development problem. However, this algorithm requires certain memory resources and computation time. The authors in [16] put forward a controller placement method based on bipartite graph for minimum weight matching. Firstly, Kuhn-Munkres algorithm is used to find the best match between switches and controllers. Then, based on genetic algorithm, a controller placement solution is found considering the average propagation delay. The authors in [17] propose two algorithms for the Resilient Capacitated Controller Placement Problem (RCCPP) in SD-WAN. Firstly, considering the capacity of the controller, the flow load of between switches and switch-controller, and the propagation delay between controllers, a resilient controller placement scheme is proposed. Secondly, according to the method of modeling NP-hard RCCPP based on the clique concept in graph theory, a polynomial time scheme is proposed. The authors in [18] propose a K self adaptive controller placement algorithm for SDWAN. The adaptive spectral clustering algorithm divides the large network into several small domains based on the spectrum clustering algorithm to maximize controller reliability and minimize WAN latency. And the structure of eigenvectors is used to automatically determine the number of SDN domains and propose metrics for spectral cluster placement. The authors in [19] consider three important indicators, including delay, hop count, and link utilization, and propose a new method to solve the controller placement problem. Based on these indicators, the analytic hierarchical process (AHP) technology is used to analyze the optimal position of the controller. In addition, an improved controller placement genetic algorithm based on this new technology is proposed to solve the position assignment problem in CPP. The above studies are all directed at the multi-controller plane in the flat control architecture without considering the hierarchical controller plane. The hierarchical architecture stratifies the control architecture to avoid the energy consumption and synchronization delay caused by frequent data synchronization between controllers in the flat architecture. The hierarchical architecture can greatly improve the scalability in SDWAN. The network partitioning in the flat architecture considers only the communication performance between the switches and the controller in that area. In the hierarchical architecture, the partitioning affects not only intra-domain communication performance but also inter-domain communication. Different from the flat control architecture, in the hierarchical control architecture, the location of root controller affects the communication performance between the root controller and the domain controller. Therefore, when the inter-domain flow arrives, the messages of communication request and business processing between the switch and the root controller will be affected, thus directly affecting inter-domain communication. Therefore, it is necessary to study in detail the network partitioning and the deployment of the root controller and the domain controller in the hierarchical control architecture.

At present, there are some studies on the hierarchical architecture. The authors in [20] propose a hybrid hierarchical control plane Orion for the increasing computational complexity in large-scale deployment network. The proposed Orion efficaciously decreases the computational complexity of the control plane in SDN. Then, aiming at the problem of path stretching caused by centralized hierarchical control plane structure, a hierarchical fast rerouting algorithm is proposed to implement fast rerouting in the proposed hybrid hierarchical control plane. Then the path stretching problem is brought about for the structure of centralized and abstracted hierarchical control plane. A hierarchical fast rerouting method is proposed to implement fast rerouting

in the proposed Orion. In order to protect the network information in each sub-network domain, the authors in [21] allocate the management of the domain controller and the root controller, thus implementing the first traffic engineering scheme of multi-domain in the hierarchical control plane. Moreover, the Network Information Base (NIB) is extended and layered, and a communication protocol is designed. The controllers at different layers and domains can allocate bandwidth cooperatively by reading the hierarchical NIB. The authors in [22] apply the Vehicular Ad-hoc Network to SDN and introduce a hierarchical distributed controller architecture, which improve the flexibility and programmability of the vehicle network. The top tier is put on the Internet, and the underlying tier is assigned at the RSU level. However, there is no detailed study on the controllers deployment in hierarchical control plane architecture. The authors in [23] propose a scheme called COLBAS to take into account the traffic load balancing problem for hierarchical networks in SDN. The COLBAS relies on the collaboration of controllers communicating through cross-controllers and designs one of the controllers as a super controller, which can flexibly manage the traffic requests processed by each controller. However, this paper only describes a hierarchical scheme, but does not specifically describe how to select domain controller and super controller, and how to classify switches. The authors in [24] propose a hierarchical multi-controller scheme for single point faults and computational complexity in large-scale networks. The proposed architecture has the potential to increase scalability and service flexibility by allocating functionality among multiple controllers organized in a hierarchy. However, this paper only focuses on the concept of hierarchical distributed control, without a detailed study of the controller location and its management domain.

In this paper, we study the controller deployment for hierarchical control plane based on Louvain algorithm. The Louvain algorithm is a community discovery algorithm based on modularity, which performs well in efficiency. The optimized goal is to maximize the modularity of the whole community network. It is able to discover the hierarchical community structure, which is in line with the hierarchical control structure. In this paper, we use the improved Louvain algorithm to partition the network into a hierarchical structure to determine the placement of the domain controller and the root controller according to the relevant performance indicators. Moreover, we formulate the modeling for request delay, reliability, and load balancing to evaluate the overall network performance for the hierarchical control architecture. Our proposed algorithm reduces a certain request delay, optimizes the load balancing among different partitions, and improves the reliability of control plane in SDWAN. Furthermore, in order to compare with our proposed algorithm, we propose improved Louvain Heuristic Algorithm [11] for promotion (ILHAP), improved k-means [25], and random deployment algorithm for hierarchical control architecture.

TABLE 1. Notations used in the paper.

Symbol	Definition
$G(V, E)$	the network topology of SDWAN, where V and E represent the sets of nodes and links , respectively.
v_i	a switch node in V
e_l	a link in E
cc	the set of domain controllers
rc	the root controller
K	the number of domain controllers, namely, the number of partitions
p_v	failure probability of node v
p_e	failure probability of link e
λ_i	the flow request rate of the switch i
$deg\,(i)$	the degree of node v_i
N_i	the number of switches in the i domain
N	total number of switches
$d\,(i, j)$	propagation delay between i and j
p	the probability that the domain controller communicates with the root controller
R_{deg_i}	the centrality index of normalized degree of node i
R_{λ_i}	the centrality index of normalized flow of node i.
ϕ_k	the maximum load tolerance of the domain controller k.
Φ	the maximum load tolerance of the root controller.

III. ALGORITHM FORMATION

A. CONTROLLER DEPLOYMENT MODEL

In this paper, we aim at the hierarchical control plane architecture, which divides the controllers into the root controller and the domain controllers. The architecture reduces the complexity for global information synchronization and improves the communication efficiency. The geographical distances between nodes are often far in SDWAN. Therefore, request delay is an important factor affecting network performance of SDWAN.

Due to the dynamic change of traffic in SDWAN, the unreasonable deployment of multiple controllers and the dynamic connection between switches and controllers, the processing capacity of controller varies greatly. It is likely to produce light load or overload of controllers, resulting in load imbalance and seriously reducing the performance in SDWAN.

The failures in SDWAN usually result in disruption of communication between the switch and the controller or between the controllers. The former will directly result in communication interruption in the data plane; the latter will affect the information synchronization between the controllers, resulting in the errors of controller decision. The high reliability in control plane can protect the control mechanism and maintain normal control in SDWAN. The optimization of reliability is mainly based on the failure rate, in most studies.

Therefore, our optimized goal is to reduce the request delay, make the load more balanced in each switch partition, and improve the reliability of the control plane.

The network topology of SDWAN is presented a weighted graph $G(V, E)$, where V is the set of nodes, and E is the set of edges. We summarize the detailed symbols and definitions used in the paper in Table 1.

1) THE REQUEST DELAY

Unlike the single controller architecture and the flat control architecture, the delay for switch to make a request in the hierarchical control architecture is divided into two parts: the delay between the switch and its domain controller and the delay between the domain controller and the root controller. For intra-domain communication, the request delay is only the delay between the switch and its domain controller, defined as follows:

$$D_{intra}(i) = d(v_i, cc_k). \tag{1}$$

where cc_k is the controller assigned to switch v_i.

For inter-domain communication, the request delay also includes the delay between the domain controller and the root controller. The request delay for inter-domain communication is defined as follows:

$$D_{inter}(i) = d(v_i, cc_k) + d(cc_k, rc). \tag{2}$$

Assuming that the probability of inter-domain communication in all communications is p, the probability of intra-domain communication is $1-p$. From (1) and (2), the request delay caused by communication between switch and controller can be expressed as:

$$\begin{aligned} D(i) &= (1-p) \cdot D_{intra}(i) + p \cdot D_{inter}(i) = d(v_i, cc_k) \\ &\quad + p \cdot d(cc_k, rc) \end{aligned} \tag{3}$$

The authors in [26], [27] state that the delay between nodes is the sum of propagation delay, processing delay, and transmission delay, and the propagation delay the main component of delay in the SDWAN. The authors in [28] define the propagation delay in SDWAN: the propagation delay between the node and the controller is proportional to the distance between the switch and the controller. Generally, the distance between nodes is very large across cities and regions in SDWAN. Therefore, in this paper, the propagation delay between nodes is considered as the delay between nodes and is defined as follows:

$$d(s, c) = Dijks(s, c) \Big/ 10^6 \tag{4}$$

where $Dijks(s, c)$ is the distance of shortest path between node s and node c calculated by Dijkstra. The metric of Dijkstra is the distance between nodes. The average request delay in SDWAN is:

$$\begin{aligned} D_{average} &= \frac{\sum_{k \in K} \sum_{i \in N_k} D(i)}{N} \\ &= \frac{\sum_{k \in K} \sum_{i \in N_k} ((1-p) \cdot D_{intra}(i) + p \cdot D_{inter}(i))}{N} \\ &= \frac{\sum_{k \in K} \sum_{i \in N_k} (d(v_i, cc_k) + p \cdot d(cc_k, rc))}{N} \end{aligned} \tag{5}$$

2) THE LOAD BALANCING INDEX

For the SDWAN control plane deployed by multiple controllers, the controller monitors and manages the partition as the core in each partition. The load of each partition managed by the controller can be regarded as the load of the controller. The ability of controller to handle the load is not only related to its own software architecture, but also restricted by the hardware resources of the server itself. The overloading of the controller leads to the increase of response delay of network events and controller failure. Therefore, the load balancing in each partition is one of the important performance indices to solve the controller deployment. Most studies only consider the balance of the number of switches in each partition. The load in each partition includes not only the number of switches in the controller management domain but also the size of the flow to be processed by each switch. In the paper, the load balancing index of the algorithm is given as following:

$$\begin{aligned} LB &= \left(\sum_{i=1}^{K} \alpha \cdot \left(\frac{\sum_{j=1}^{N_i} \lambda_{i,j} - \frac{\sum_{i=1}^{K} \sum_{j=1}^{N_i} \lambda_{i,j}}{K}}{\sum_{i=1}^{K} \sum_{j=1}^{N_i} \lambda_{i,j}} \right)^2 + \beta \cdot \left(\frac{N_i - \frac{N}{K}}{N} \right)^2 \right)^{1/2} \\ &= \left(\sum_{i=1}^{K} \alpha \cdot \left(\frac{\sum_{j=1}^{N_i} \lambda_{i,j}}{\sum_{i=1}^{K} \sum_{j=1}^{N_i} \lambda_{i,j}} - \frac{1}{K} \right)^2 + \beta \cdot \left(\frac{N_i}{N} - \frac{1}{K} \right)^2 \right)^{1/2} \end{aligned} \tag{6}$$

where $\lambda_{i,j}$ represents the flow request rate of the switch j in the domain i and $\alpha+\beta=1$. α and β respectively represent the weighting coefficients in the load balancing. When $\alpha=1$, it means that the load balancing index is simply considered from flow, which shows the load balancing degree of the whole network. When $\beta=1$, it means that each partition is balanced only from the number of switches.

When partitioning the network, we define the objective function as:

$$min \quad LB \tag{7}$$

The constraint is

$$\vec{f}_{v_i} = \overleftarrow{f}_{v_i}, \quad v_i \in V \tag{8}$$

$$l_{i,j} \leq 1, \quad (i,j) \in E \tag{9}$$

$$\sum_{i \in N_i} \lambda_{i,k} \leq \phi_k, \quad k \in K \tag{10}$$

The equation (8) indicates that the inflow and outflow of a switch are equal. The inequation (9) ensures that the link utilization of each link is less than 1. The inequation (10) denotes that the total load of each domain cannot exceed the maximum load tolerance of the domain controller.

In the distributed multi-controller deployment scenario, the reliability of the control plane is an important performance parameter. The higher the reliability, the longer time the control plane continues to work without failure. The failures usually result in communication interruption between the domain controller and the switch or between the domain controller and the root controller. Therefore, the reliability is also one of the important performance indices to solve the controller deployment, focusing on the reliability of control path in most studies. Reliability includes the reliability of the nodes and the links. The links include the links between the switch and the domain controller, and between the domain controller and the root controller.

In this paper, we do not focus on the design of routing algorithms, so the communication path from switch to controller is calculated by the Dijkstra algorithm, where the weight between the nodes is considered to be the geographical distance. The path reliability between node i and node j is defined as:

$$R(i,j) = \prod_{e \in E_{i,j}} (1 - p_e) \prod_{v \in V_{i,j}} (1 - p_v) \tag{11}$$

where $E_{i,j}$ represents the links set of the shortest path between node i and node j, and $V_{i,j}$ represents the nodes set of the shortest path between node i and node j.

We assume that the kth domain controller is deployed at node $cc(k)$ and the root controller is deployed at node rc. We consider the sum of the path reliabilities of the domain controller and the all switches in its administrative domain as the path reliability. So the path reliability of the domain controller in the kth partition is as follows:

$$\begin{aligned} R_c(k) &= \sum_{j \in C_k} R(cc(k), j) \\ &= \sum_{j \in C_k} \prod_{e \in E_{cc(k),j}} (1 - p_e) \prod_{v \in V_{cc(k),j}} (1 - p_v) \end{aligned} \tag{12}$$

where C_k denotes the kth partition where node j is located.

We treat the sum of the path reliabilities between the root controller and all domain controllers as the path reliability of the root controller. The path reliability of the root controller is:

$$\begin{aligned} R_{rc} &= \sum_{k \in K} R_c(k) \cdot R(cc(k), rc) \\ &= \sum_{k \in K} \sum_{j \in C_k} R(cc(k), j) \cdot R(cc(k), rc) \\ &= \sum_{k \in K} \sum_{j \in C_k} \prod_{e \in E_{cc(k),j}} (1 - p_e) \cdot \prod_{v \in V_{cc(k),j}} (1 - p_v) \\ &\quad \cdot \prod_{e \in E_{cc(k),rc}} (1 - p_e) \cdot \prod_{v \in V_{cc(k),rc}} (1 - p_v) \end{aligned} \tag{13}$$

Many studies only consider the path reliability in solving reliability problems, but do not consider other factors of nodes. In this paper, we consider not only the reliability of nodes and links, but also the flow request rate and degree of the node. The degree of a node is the number of edges associated with the node. The greater the degree of the node, the greater the role and influence of the node in the network. And the higher the flow request rate of the node, the stronger the ability of the node to process the flow.

In order to compare the influence of nodes in the network, the centrality index of normalized degree of node i is defined as:

$$R_{degi} = \frac{\deg(i)}{\max\limits_{i \in N}(\deg(i))} \tag{14}$$

Similarly, in order to compare the flow processing capabilities of nodes in the network, the centrality index of normalized flow of node i is defined as:

$$R_{\lambda_i} = \frac{\lambda_i}{\max\limits_{i \in N}(\lambda_i)} \tag{15}$$

We jointly consider the three factors of reliability of nodes and links, flow request rate, and degree of node to define the reliability of the domain controller node and the root controller node. If node i is deployed as a domain controller in the kth partition, the reliability of the node is defined as:

$$R_i^c = R_c(k) \cdot R_{degi} \cdot R_{\lambda_i} \tag{16}$$

If node i is deployed as a root controller, the reliability of the node is defined as:

$$R_i^{rc} = R_{rc} \cdot R_{degi} \cdot R_{\lambda_i} \tag{17}$$

When selecting the root controller and domain controller, we define the objective function as:

$$max \ R_i^c \tag{18}$$

$$max \ R_i^{rc} \tag{19}$$

The constraint is

$$\sum_{i \in N_i} \lambda_{i,k} \leq \phi_k, \quad k \in K \tag{20}$$

$$\sum_{k \in K} \sum_{i \in N_i} \lambda_{i,k} \leq \Phi \tag{21}$$

$$\sum y_{cc}^{rc} = K \tag{22}$$

The inequation (20) indicates the total load of each domain less than the maximum load tolerance of the domain controller. The inequation (21) shows the total load of all inter-domains in the whole network is less than the maximum load that the root controller can bear. The equation (22) ensures that all domain controllers are linked to the root controller.

B. THE LOUVAIN ALGORITHM

The Louvain algorithm is an modularity optimization algorithm based on multi-level (round-by-round heuristic iteration). The hierarchical community structure is found through the following calculation processes. The algorithm can be divided into two phases and iterated repeatedly. The Louvain

algorithm is based on modularity. The modularity is defined as:

$$Q=\frac{1}{2m}\sum_{i,j}\left[A_{ij}-\frac{W_iW_j}{2m}\right]\delta(C_i,C_j) \tag{23}$$

where A_{ij} denotes the weight of the nodes i and j and W_i is the sum of the weights of the edges connected to node i. C_i represents the community to which i belongs. $\delta(u,v)$ denotes whether u and v are the same community. If u and v are the same community, this value is 1, otherwise 0.

$$m=\frac{1}{2}\sum_{ij}A_{i,j} \tag{24}$$

The nodes in the network are taken out from the original community and continuously traversed. We calculate the modularity gain generated by this node being added to each community. Then we choose the community with the largest gain in modularity, add the node to it until no node can be moved, and merge the community into a super node. Repeat the above steps until the modularity is no longer increased. The modularity gain is the change of modularity when a node is taken out from the original community and added to another community. The formula for calculating modularity gain is as follows:

$$\begin{aligned}\Delta Q=&\left[\frac{\sum_{in}+w_{i,in}}{2m}-\left(\frac{\sum_{tot}+w_i}{2m}\right)^2\right]\\&-\left[\frac{\sum_{in}}{2m}-\left(\frac{\sum_{tot}}{2m}\right)^2-\left(\frac{w_i}{2m}\right)^2\right]\\=&\frac{1}{2m}\left(w_{i,in}-\frac{\sum_{tot}w_i}{m}\right)\end{aligned} \tag{25}$$

where $w_{i,in}/m$ denotes the impact of putting isolated nodes and community C together on the modularity in the entire network and $\sum_{tot} w_i/m^2$ represents the influence of isolated nodes and community C on the modularity in the whole network , so their difference reflects the influence of isolated nodes on the modularity of whole network before and after they are put into community C.

The first phase of the algorithm is completed by repeating the process until all nodes no longer move. If the modularity in the whole network does not change after one iteration, the iteration is stopped. If the modularity has not reached the local optimum, the iteration continues. When the first phase stops counting, the local modularity reaches the maximum value. In the second phase of the algorithm, a new network is built by the communities discovered in the first phase. Each community is regarded as a new node, and the nodes in the community are compressed. The weight between the new nodes is determined by the sum of the weights between the nodes in the community. When the new network is established, the process of the first phase is repeated until the community with largest local modularity is obtained. The flow chart of the Louvain algorithm is shown in Fig.2.

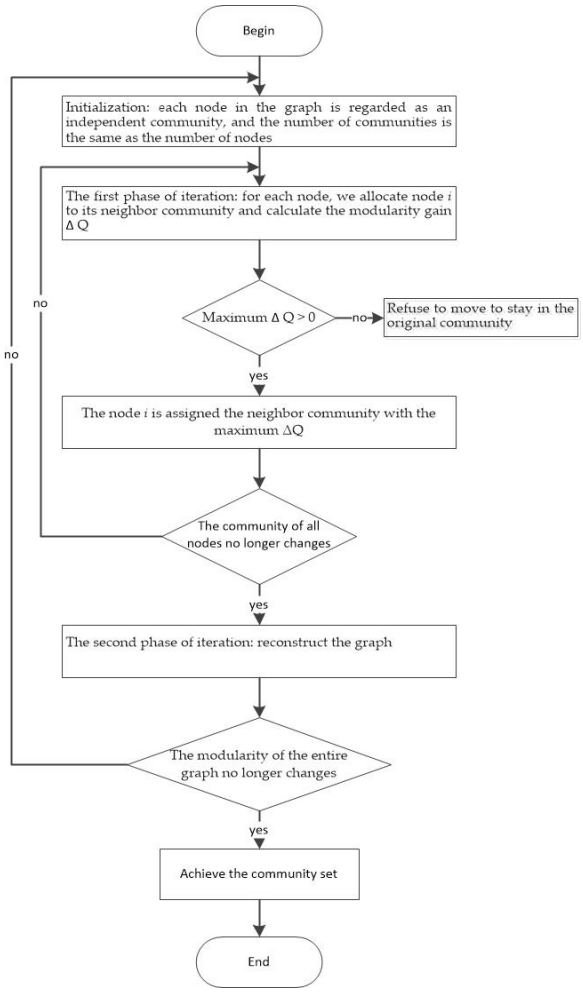

FIGURE 2. Flow chart of Louvain algorithm.

C. PROPOSED SOLUTION

In the hierarchical control architecture in SDWAN, the distance between switches and domain controllers, as well as between domain controllers and root controller, is relatively distant. Therefore, delay is an important factor affecting network performance. As the network scale increases, the load balancing technology can not only maintain the balanced distribution of load in the network, but also maintain the high availability of the network. So in this paper, we consider delay and load balancing to optimize the modularity when the network is partitioned.

At present, the reliability is not well considered for controller placement in hierarchical network architecture. However, the link failure between the controller and the switch will cause the failure of flow forwarding, which prevents the switch from receiving instructions from the controller. Moreover, the failure between the root controller and the domain controller will lead to the domain controller to fail to accept the control messages from its upper controller, thus affecting the flow forwarding between switches. In this paper, we study the placement of root controller and domain controllers in hierarchical architecture to optimize network reliability.

TABLE 2. The pseudo code of improved Louvain algorithm.

Algorithm: improved Louvain algorithm
1.Input: $G(V, E)$, delay matrix between nodes D, the set of flows request rate for all the switches $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_N)$.
2. Output: the partition C
3. $t = 1$
4. while $modularity\ change$ do
5. while $node\ in\ the\ community\ moves$ do
6. each node is considered a community
7. for each node i
8. node i is added to the community of the node i's neighbors in turn.
9. if $t == 1$
10. the weight $A_{i,j}$ is represented by the propagation delay $d(i, j)$ and ΔQ is calculated by (16).
11. else
12. $\Delta Q = LB - LB'$ (LB denotes the modularity of the whole network before node i migration and LB denotes the modularity of the whole network after node i migration. In this case, modularity represents load balance index)
13. end
14. if $\max(\Delta Q) > 0$
15. add node i to the community where the neighbor node with the maximum modularity is located.
16. else
17. node i is still in the original community.
18. end
19. end
20. All nodes in the same community are compressed into a new node.
21. $t = t + 1$
22. end

The proposed algorithm is divided into two parts: the allocation of the communities and the selection of the controllers (domain controllers and root controller). The modularity function of Louvain algorithm can not only describe the closeness of the community, but also be used as an optimization function. If the modularity of the current community structure can be improved by adding a node to the community of its neighbor, then this iterative optimization is acceptable. In the first part of the proposed algorithm, we partition the network topology through the improved Louvain algorithm by modifying the modularity. In the first phase of the first iteration of the improved Louvain algorithm, the weight between nodes is represented by the reciprocal of propagation delay between nodes. The smaller the propagation delay between nodes, the larger the weights, the easier the nodes can be divided into the same partition. And the whole network is divided into small partitions with delay as the optimized objective. The first partition of the network is realized. The detailed steps are shown in Table 2. In the process of super node merging, the load balancing index of the whole network is regarded as the modularity. The smaller the load balancing index, the more balanced the network. Therefore, we consider the difference between the modularity of the whole network before node *i* migration and after node *i* migration. If the maximum modularity gain is greater than 0, the node should move to the neighbor community with the maximum modularity. When the modularity reaches the minimum and no longer changes, the partition reaches stability and the load balancing of the whole network is achieved optimally. The deployment algorithm for the domain controller and root controller is shown in Table 3. The improved Louvain algorithm in Table 2 and controller development algorithm in Table 3 constitute our proposed algorithm.

TABLE 3. The pseudo code for deploying domain controllers and root controller.

Algorithm: deployment algorithm of domain controllers and root controller
1.Input: the partition C, the number of partitions K, the set of the reliability of the nodes R^c, R^{rc}
2. Output: the deployment location C_{domain},C_{root}
3. for $i = 1 : K$
4. for $j = 1 : \lvert C_i \rvert$
5. $RR^c_{v_j} = \sum_{s \in C_i / v_j} R^c_s$
6. end
7. $C_{domain}(i) = \arg\max_{v_j} RR^c_{v_j}$
8. end
9. for $i = 1 : \lvert C_{domain} \rvert$
10. $R_c(i) = \sum_{j \in C_{domain}/i} R^{rc}_j$
11. end
12. $C_{root} = \arg\max_i R_c$

D. ALGORITHM SUMMARY AND COMPLEXITY ANALYSIS

The proposed algorithm is divided into two parts: community partitioning and deployment of domain controllers and root controller. We partition the network by the improved Louvain algorithm. The improved Louvain algorithm includes two phases: partitioning the nodes and reconstructing the network. In the first phase of the first iteration, we divide the network into multiple small partitions with delay as the optimized objective. The nodes in the same partition are compressed into a super node. Then, with load balancing as the optimized goal, the new network composed of super nodes is partitioned and reconstructed until the modularity no longer changes. After partitioning the network, the node with the greatest reliability is deployed as the domain controller in each partition, and then the domain controller node with the highest reliability is placed as the root controller.

In the first phase of the improved Louvain algorithm, the algorithm traverses all neighbor nodes of each node, and measures the modularity gain brought by adding the node to the community where the neighbor node is located. So the time complexity in each iteration is $O(|E|)$, and $|E|$ is the number of edges in the network topology. In the second phase of the algorithm, the algorithm folds each community into a single node, calculates the joint weight between these newly generated community nodes, and the sum of the joint weights between all nodes in the community. So, the time complexity of the second phase is $O(|E| + N')$, and N' is the number of nodes in current iteration. So the time complexity of the improved Louvain algorithm is $O(2 * |E| + N')$. The deployment complexity of a domain controller is related to the number of switches in the domain. The deployment complexity of the root controller depends on the number of domain controllers. The time complexity of domain controllers deployment and root controller deployment is $O(N)$ and $O(K)$ respectively. N is the number of all switch nodes in the network, and K is the number of network partitions. Therefore, the time complexity of the whole algorithm is $O(2 * |E| + N' + N + K)$. Because of the number of partitions $K << N$, we can consider the time complexity of the proposed algorithm is $O(|E| + N' + N)$. The number of the nodes

and links in the network can directly affect the complexity of the algorithm. N' depends on the partitioning effect of the algorithm, and the better the partitioning effect, the smaller the N'. The partitioning effect of the algorithm depends on the strategy of weight calculation. So the weight calculation method needs to be well studied so that the algorithm can partition the network well, so as to reduce the complexity of the algorithm.

IV. PERFORMANCE EVALUATION

A. DESCRIPTION OF THE COMPARED ALGORITHM

Most papers on hierarchical architecture focus on the functional design of hierarchical architecture, with little mention of network partitioning and controller placement for hierarchical architecture. In order to compare with our proposed algorithm, we improve three traditional algorithms to solve the network partitioning and controller placement for hierarchical architecture. The authors in [11] present an improved Louvain Heuristic Algorithm (ILHA) to partition the network, and then select the controller location based on the minimum maximum delay or the minimum average delay. In order to make a better comparison with our algorithm, we add the step of selecting the root controller with the maximum reliability on the basis of ILHA algorithm, as shown in lines 6-12 in Table 3. In the paper, we select the domain controller location based on the minimum average delay and call it improved Louvain Heuristic Algorithm for promotion (ILHAP). Then we improve the k-means algorithm for hierarchical architecture as the second comparison algorithm. The k-means algorithm is a distance-based clustering algorithm. In this paper, we consider the distance to be a reflection of the delay. The steps of the improved k-means algorithm are as follows. Firstly, given the number of partitions K, the k-means algorithm divides the network into k partitions based on the distance between the nodes. Then, the centroid in each partition is deployed as the domain controller. Finally, the domain controller node with the smallest sum of distances from other domain controllers nodes is deployed as the root controller. And we propose a random deployment algorithm for hierarchical architecture as the third comparison algorithm. The steps of the random deployment algorithm are as follows. First, according to the given number of partitions K, the network is randomly divided into K partitions. Then, the location of the domain controller is randomly deployed in each partition. Finally, the location of the root controller is selected randomly from the deployment location of the domain controller.

B. SIMULATION SETTINGS

In this section, we compare the performance of our proposed algorithm with ILHAP algorithm, the improved k-means, and random deployment algorithm for hierarchical architecture. We use MATLAB to simulate and take the average result of each algorithm performed 100 times. And we use Internet2 OS3E [29] network topology and other four topologies with different scales from Internet Topology Zoo [30] to evaluate the performance of the proposed algorithm. The dimensions of five network topologies are shown in Table 4.

TABLE 4. The dimensions of five network topologies.

Network topology	The number of nodes	The number of links
Aranet	19	24
Abvt	23	31
Arpanet19728	29	32
Internet2 OS3E	34	42
Carnet	44	43

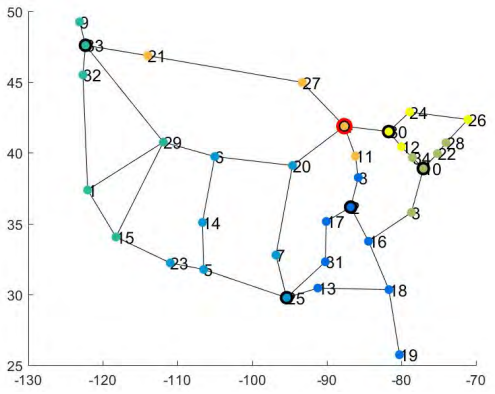

FIGURE 3. The result of partitioning and controller deployment of our proposed algorithm for Internet2 OS3E.

C. PERFORMANCE COMPARISON AT THE SAME NUMBERS OF PARTITIONS

In this section, we compare the performance of our proposed algorithm with ILHAP algorithm, the improved k-means algorithm, and the random deployment algorithm in terms of load balancing, reliability, and average request delay. The proposed algorithm and ILHAP algorithm can adaptively calculate the number of partitions according to the network topology with optimal modularity. The improved k-means algorithm and random deployment algorithm need to specify the number of partitions in advance. In order to make a better comparison under the same number of partitions, we set the number of partitions for the improved k-means algorithm and the random deployment algorithm to be the same as the number of partitions in our proposed algorithm and ILHAP algorithm in Fig. 4 to Fig. 6. The proposed algorithm is divided into two parts: network partitioning and controllers placement. We use the improved Louvain algorithm to partition the network topology. Fig. 3 takes Internet2 OS3E as an example to show the result of partitioning and controller deployment of our proposed algorithm. From the Fig. 3, we can see that the improved Louvain algorithm divides the Internet2 OS3E topology into six partitions. The different colors represent different partitions, and the nodes of the same color are represented in the same partition. The nodes with a black circle indicate the deployment nodes of domain controllers. The node with a red circle denotes the deployment node of root controller. The experimental result shows that the ILHAP algorithm also divides the Internet2 OS3E topology into six parts.

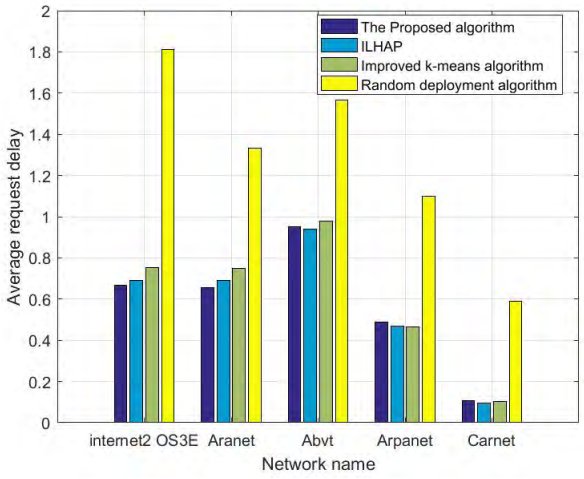

FIGURE 4. The average request delay of the four algorithms in different networks.

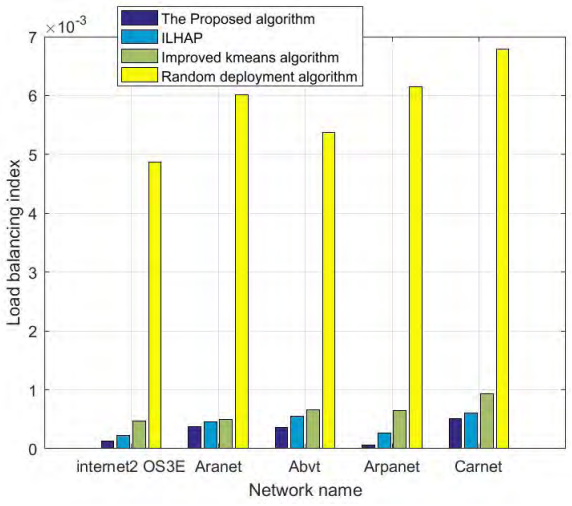

FIGURE 5. The load balancing index of the four algorithms in different networks when $\alpha = 0.5, \beta = 0.5$.

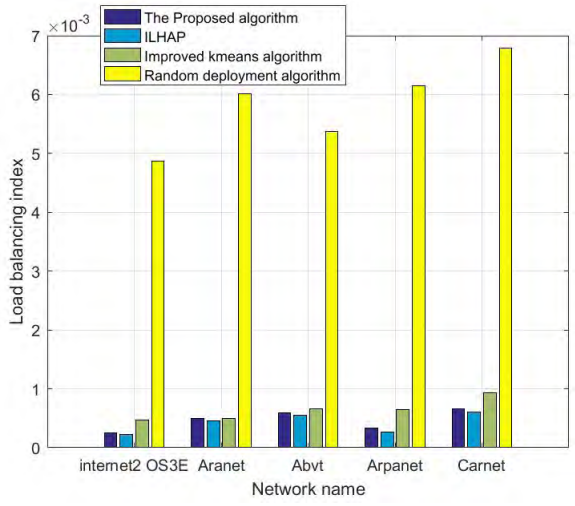

FIGURE 6. The load balancing index of the four algorithms in different networks when $\alpha = 1, \beta = 0$.

Fig. 4 shows the average request delay of four algorithms in different networks. As shown in Fig. 4, the average request delay of the random deployment algorithm is largest. Because the random deployment algorithm does not take into account the performance parameters in the network, but adopts a random selection method. In the Internet2 OS3E, Aranet, and Abvt networks, the average request delay of the proposed algorithm is smaller than the improved k-means algorithm. The average request delay of improved k-means algorithm in Arpanet network is smallest. In Carnet network, the average request delay of the proposed algorithm and k-means algorithm is basically the same. Because k-means algorithm is an iterative algorithm based on distance, the distance between each node and its domain controller is smaller than other domain controllers. And the sum of the distances from the root controller to other domain controllers is the shortest. The improved k-means algorithm regards distance as the only optimized objective and the distance reflects the delay in SDWAN. The average request delay of ILHAP algorithm is larger than our proposed in Internet2 OS3E and Aranet, and less than the proposed algorithm in the other three networks. The weight between two nodes is related to distance in the ILHAP algorithm. So the average request delay of ILHAP algorithm can be effectively reduced. In the initial iteration, our proposed algorithm divides the network with the delay as the optimized objective, but the delay is not the only optimized goal. Therefore, we can conclude that our algorithm can reduce the average request delay to some extent.

Fig. 5 shows the load balancing indices of four algorithms in different networks when $\alpha = 0.5, \beta = 0.5$. The load balancing index represents the performance of load balancing in SDWAN. The smaller the load balancing index, the more balanced the load in the network. We can see that the load balancing index of the random deployment algorithm is the largest, and the load balancing performance is the worst in the five network topologies. Because the random deployment algorithm is deployed randomly in partitions and the selection of domain controllers and root controller. The load balancing performance of the improved k-means algorithm is better than the random deployment algorithm. The load balancing index of the ILHAP algorithm is larger than the proposed algorithm. The ILHAP algorithm only considers the influence coefficient of the node in the load balancing index, and limits the difference of the total influence coefficient between the two communities to the β. The load balancing index of our proposed algorithm is the lowest and the performance for load balancing is best compared to the other three algorithms. When $\alpha = 0.5, \beta = 0.5$, our proposed algorithm considers the balance from the number of nodes and the request rate of flows for each node. Therefore, our proposed algorithm achieves better performance for load balancing.

Fig. 6 shows the load balancing indices of four algorithms in different networks when $\alpha = 1, \beta = 0$. The load balancing index of the proposed algorithm is simply considered the request rate of flows for each node when $\alpha = 1, \beta = 0$. The larger the flow request rate of the node, the larger the

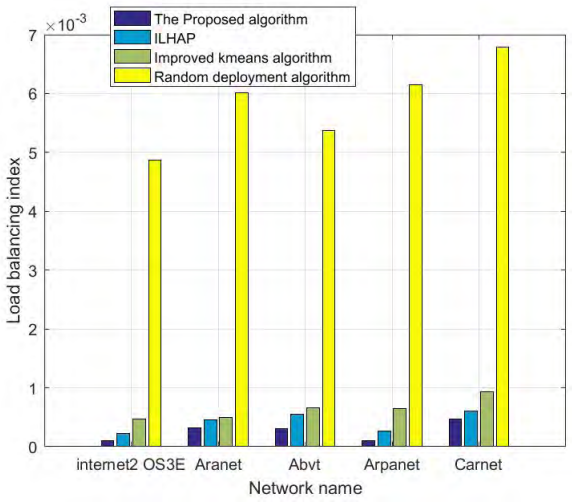

FIGURE 7. The load balancing index of the four algorithms in different networks when $\alpha = 0, \beta = 1$.

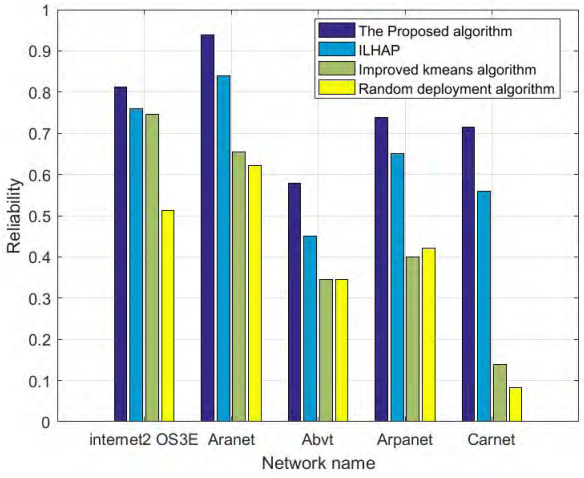

FIGURE 8. The reliability of the four algorithms in different networks.

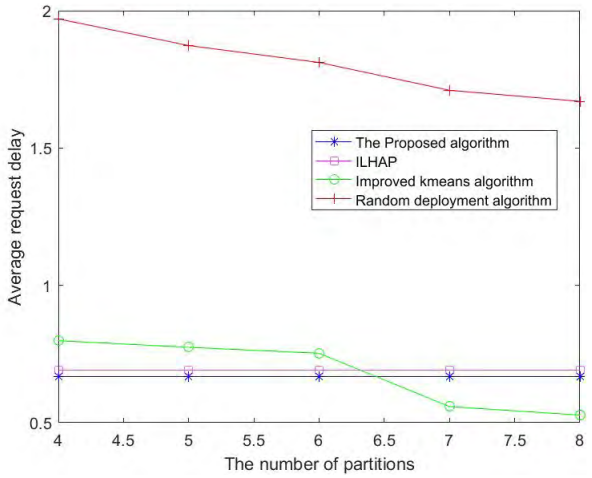

FIGURE 9. The average delay at different numbers of partitions in Internet2 OS3E network.

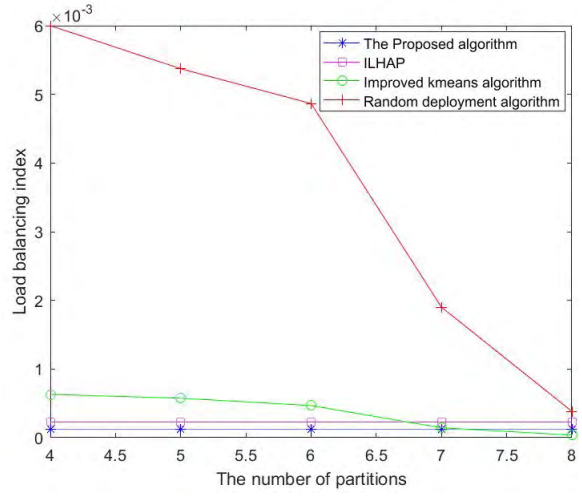

FIGURE 10. The load balancing index at different numbers of partitions in Internet2 OS3E network when $\alpha = 0.5, \beta = 0.5$.

influence factor of the node on the network. The performance of the proposed algorithm is worse than that of ILHAP algorithm when load balancing is evaluated only from the impact factor of nodes.

Fig. 7 shows the load balancing performance when $\alpha = 0, \beta = 1$. The load balancing performance of the proposed algorithm is only considered in terms of the number of nodes when $\alpha = 0, \beta = 1$. The load balancing performance of ILHAP algorithm is considered from the influence coefficient of nodes. So the load balancing performance of the ILHAP algorithm is worse than that of the proposed algorithm in Fig. 7.

Fig. 8 shows the reliability of the four algorithms in different networks. It can be seen from the Fig. 8 that our proposed algorithm has great advantages in reliability compared with the ILHAP algorithm, improved k-means algorithm, and the random deployment algorithm. The ILHA algorithm does not consider the deployment of the root controller nor the reliability, so we add the root controller deployment to the most reliability location in the ILHAP algorithm. The proposed algorithm takes into account not only the path reliability between nodes but also the degree and flow request rate of node when deploying the domain controller and root controller. The greater the reliability of controller deployment node, the longer the duration of trouble-free operation in SDWAN. The proposed algorithm improves the reliability and availability of the control plane in the hierarchical architecture.

D. PERFORMANCE COMPARISON AT DIFFERENT NUMBERS OF PARTITIONS

The improved Louvain algorithm is a modularity-based community discovery algorithm. It does not need to specify the number of partitions in advance and can discover hierarchical community structure. When the modularity of the whole network is optimal, the algorithm stops iterating. Taking Internet 2 OS3E as an example, in the proposed algorithm and ILHAP algorithm, the network is divided into

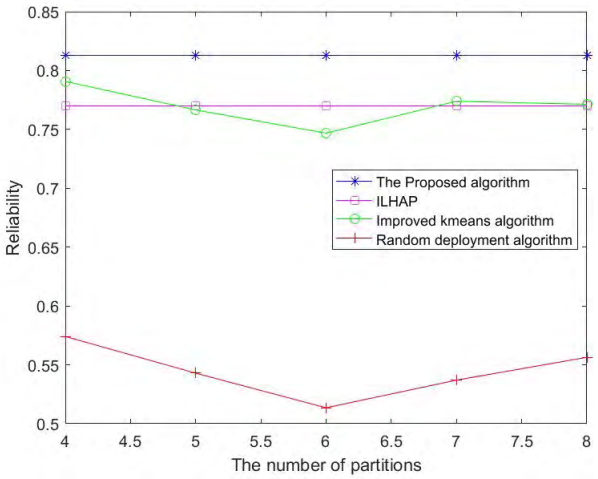

FIGURE 11. The reliability at different numbers of partitions in Internet2 OS3E network.

six partitions. The improved k-means algorithm and random deployment algorithm need to specify the number of partitions. In Fig. 4 to Fig. 8, we compare the performance of the four algorithms in different networks when the number of partitions for improved k-means algorithm and the random deployment algorithm is equal to the number of partitions for our proposed algorithm and ILHAP algorithm. In Fig. 9 to Fig. 11, we take the Internet 2 OS3E network as an example to compare the performance of the ILHAP algorithm, improved k-means algorithm, and the random deployment algorithm under different partitions with our proposed algorithm. The proposed algorithm and ILHAP algorithm have fixed number of partitions, so their performance is represented by straight line in Fig. 9 to Fig. 11, respectively. We consider the performance of the improved k-means algorithm and random deployment algorithm when the number of partitions is 4,5,6,7,8.

Fig. 9 and Fig. 10 show the average request delay and load balancing at different numbers of partitions in Internet2 OS3E network. We can see that as the number of partitions increases, the average request delay and load balancing index of improved k-means and random deployment algorithms become smaller and smaller. As the network is divided into more partitions, the number of switches in each partition is smaller and the average request latency and load balancing index is smaller. The average request delay and load balancing performance of random deployment algorithm is worst compared to the improved k-means algorithm, ILHAP algorithm, and proposed algorithm as the number of partitions changes. When the number of partitions is greater than six, the average request delay of improved k-means algorithm is lower than the ILHAP algorithm and our proposed algorithm. When the number of partitions is greater than seven, the load balancing index of improved k-means algorithm is lower than the ILHAP algorithm and our proposed algorithm. As we all know, the more partitions, the more domain controllers need to be deployed, the greater the cost of network deployment.

TABLE 5. The average running time of different algorithms to execute once.

Algorithms	The proposed algorithm	ILHAP algorithm	Improved kmeans algorithm	Random deployment algorithm
running time/s	0.6126	1.0813	1.1220	0.0418

The improved Louvain algorithm can be partitioned reasonably according to topology and performance of network.

Fig. 11 shows the reliability at different numbers of partitions in Internet2 OS3E network. As the number of partitions increases, the reliability of the random deployment algorithm is the worst. The reliability of the improved k-means algorithm is greater than the random deployment algorithm. When the number of partitions is five or six, the reliability of ILHAP algorithm is greater than the improved k-means algorithm. The proposed algorithm has the highest reliability. Reliability is related to the failure of links and nodes as well as the degree and flow request rate of node, and has little to do with the number of partitions.

We compare the average running time of the four algorithms in Table 5. We can see that the running time of the proposed algorithm is less than the ILHAP algorithm and the improve k-means algorithm. And the average running time of the proposed algorithm is almost half of the improved k-means algorithm. The convergence rate of ILHAP algorithm is slower when the constraint of influence coefficient is not satisfied. In the proposed algorithm, the network topology is divided into multiple small partitions by the improved Louvain algorithm. Each partition is regards as a super node and network is compressed. It greatly reduces the number of edges and nodes. But the random deployment algorithm has the shortest average running time. Because random deployment algorithm uses random selection method to select network partitioning and controller deployment nodes, without considering network performance. Although the running time of our proposed algorithm is longer than the random deployment algorithm, the deployment is done once when the network is originally built, so it does not have the obvious disadvantages. In general, the proposed algorithm has a good effect on the execution efficiency.

V. CONCLUSIONS

In this paper, we propose a multi-controller deployment algorithm in hierarchical architecture for SDWAN. We propose an evaluation modeling for request delay, load balancing, and reliability for the hierarchical control architecture. Firstly, in order to optimize the delay and load balancing among different partitions, we put forward an improved Louvain algorithm for network partitioning. We consider the reliability and availability in hierarchical control plane to deploy domain controllers and root controller for SDWAN. For load balancing, we consider the balance of the number of switches and the size of the flows in each partition. We consider the reliability of hierarchical control architecture in terms of path reliability, degree and flow request rate of node. The simulation

results show that the proposed algorithm reduces the average request delay to a certain extent in SDWAN, optimizes the load balancing among different partitions, and improves the reliability of the control plane. And the proposed algorithm achieves good effect on the execution efficiency. Since there are still many factors to be considered in actual deployment, our future research will consider deployment costs within the community, delay at the control plane and so on.

REFERENCES

[1] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] F. N. N. Farias, J. J. Salvatti, E. C. Cerqueira, and A. J. G. Abelém, "A proposal management of the legacy network environment using OpenFlow control plane," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 1143–1150.

[3] J. Hu, C. Lin, X. Li, and J. Huang, "Scalability of control planes for software defined networks: Modeling and evaluation," in *Proc. IEEE 22nd Int. Symp. Qual. Service (IWQoS)*, May 2014, pp. 147–152.

[4] R. Veisllari, N. Stol, S. Bjornstad, and C. Raffaelli, "Scalability analysis of SDN-controlled optical ring MAN with hybrid traffic," in *Proc. IEEE Int. Conf. Commun.(ICC)*, Jun. 2014, pp. 3283–3288.

[5] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 19–24.

[6] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, Apr. 2010, pp. 1–6.

[7] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. 9th USENIX Conf. Operating Syst. design Implement.*, Vancouver, Canada, Oct. 2010, pp. 1–6.

[8] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Aug. 2014, pp. 1–6.

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 473–478, Sep. 2012.

[10] J. Liu, J. Liu, and R. Xie, "Reliability-based controller placement algorithm in software defined networking," *Comput. Sci. Inf. Syst.*, vol. 13, no. 2, pp. 547–560, 2016.

[11] C. Wen, C. Cong, J. Xueqin, and L. Leijie, "Multi-controller placement towards SDN based on Louvain heuristic algorithm," *IEEE ACCESS*, vol. 6, pp. 49486–49497, 2018.

[12] A. Sallahi and M. St-Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 274–277, Feb. 2017.

[13] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[14] V. Ahmadi and M. Khorramizadeh, "An adaptive heuristic for multi-objective controller placement in software-defined networks," *Comput. Elect. Eng.*, vol. 66, pp. 204–228, Feb. 2018.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[16] Y. Tingting, H. Xiaohong, M. Maode, and Y. Jie, "Balance-based SDN controller placement and assignment with minimum weight matching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[17] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 991–1005, Sep. 2018.

[18] P. Xiao, Z.-Y. Li, S. Guo, H. Qi, W.-Y. Qu, and H.-S. Yu, "A K self-adaptive SDN controller placement for wide area networks," *Frontiers Inf. Technol. Electron. Eng.*, vol. 17, no. 7, pp. 620–633, Jul. 2016.

[19] A. Jalili, M. Keshtgari, R. Akbari, and R. Javidan, "Multi criteria analysis of controller placement problem in software defined networks," *Comput. Commun.*, vol. 133, pp. 115–128, Jan. 2019.

[20] Y. Fu *et al.*, "A hybrid hierarchical control plane for flow-based large-scale software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 2, pp. 117–131, Jun. 2015.

[21] H. Jingyu, Z. Laiping, Z. Suohao, L. Yangyang, G. Xin, and Z. Sheng, "Topology-preserving traffic engineering for hierarchical multi-domain SDN," *Comput. Netw.*, vol. 140, pp. 62–77, May 2018.

[22] K. S. K. Liyanage, M. Ma, and P. H. J. Chong, "Controller placement optimization in hierarchical distributed software defined vehicular networks," *Comput. Netw.*, vol. 135, pp. 226–239, Apr. 2018.

[23] H. Selvi, G. Gür, and F. Alagöz, "Cooperative load balancing for hierarchical SDN controllers," in *Proc. IEEE 17th Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2016, pp. 100–105.

[24] P. D. Bhole and D. D. Puri, "Distributed hierarchical control plane of software defined networking," in *Proc. Int. Conf. Comput. Intell. Commun. Netw.*, Dec. 2015, pp. 516–522.

[25] R. Z. Krista, "An efficient k'-means clustering algorithm," *Pattern Recognit. Lett.*, vol. 29, no. 9, pp. 1385–1391, Jul. 2008.

[26] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, Jan. 2015.

[27] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.

[28] K. S. Sahoo *et al.*, "On the placement of controllers in software-defined-WAN using meta-heuristic approach," *J. Syst. Softw.*, vol. 145, pp. 180–194, Nov. 2018.

[29] (2010). *Internet2 Open Science, Scholarship and Services Exchange*. [Online]. Available: http://www.internet2.edu/network/ose/

[30] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

HOU XIAOLAN was born in 1988. She received the master's degree in communications engineering from Yanshan University, Hebei, China, in 2016. She is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications (BUPT). Her research interest includes controller placement for software-defined networks.

WU MUQING was born in 1963. He received the Ph.D. degree. He is currently a Professor with the Beijing University of Posts and Telecommunications (BUPT). His research interests include new network architecture, communication networks, high-speed network traffic control and performance analysis, and GPS locating and services. He is a Senior Member of the China Institute of Communications.

LV BO received the Ph.D. degree in information and communication from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014. He was a Visiting Scholar with the Department of Computer Science, Worcester Polytechnic Institute (WPI), Worcester, MA, USA, in 2016. He is currently a Researcher with the Innovation Center, China Academy of Electronics and Information Technology. His research interests include urban computing and data visualization.

LIU YIFENG received the Ph.D. degree in electronic engineering from Wuhan University, Wuhan, China, in 2016. He is currently the Principal Investigator of machine intelligence with the Innovation Center, China Academy of Electronics and Information Technology, Beijing, China. His current research interests include machine learning and computer vision.