Heuristics for SDN Controller Deployment using Community Detection Algorithm

Bangzhou Liu, Binqiang Wang and Xiaoqiang Xi
National Digital Switching Engineering & Technological Research Center
Zhengzhou, Henan Province, China
liubangzhou@163.com

Abstract—**This paper defines several metrics to improve network performance, and proposes an approach of two steps to partition domains and deploy controllers for large SDN, reducing the complexity of multiple controller placement model compared with previous works. Firstly, the approach divides the network into several domains based on community detection algorithms. Then, the controllers are deployed in the domains separately considering control link transmission latency, reliability and controller load balance. Results show that this approach improves the load balance drastically compared with LPA, as well as network performance compared with CCP. In Internet2, the reliability index increases by 10% and the control link transmission latency decreases by 9% at most.**

Keywords-software-defined networks (SDN); domain partition; controller deployment

I. INTRODUCTION

Software-defined networks[1] largely changes the design and management of past network, whose separation of control logic and data forwarding makes it easier for more fine-grained network management and resource scheduling. With the continuous development of related technologies, application of SDN technology is no longer confined to the campus network, data center network and other small networks, and larger SDN has become a research hotspot. Google has applied SDN to connect global data center backbone link to improve network resource utilization. Telecom operators use SDN to achieve flexible scheduling WAN traffic between MAN and IDC private network, enhancing the efficiency of the network operation and maintenance.

Typical large-scale SDN often use flat control architecture for distributed control[2]. Controllers deployed in different domains, though relatively geographically dispersed, are logically centralized upward through information exchange. How to maintain a logical view of the whole network easily, ensure the reliability of the control plane and enhance nodes connectivity by deploying controllers has become a major challenge SDN control plane design.

In recent years, multi-controller deployment becomes hot. [3]models this problem as location problems. In considering the case of a control link transmission latency, required number and location of the controller deployment were discussed. [4] proposed controller deployment integrated cost model, which considers controller deployment overhead cost, communication overhead between the controller and the switch and connectivity overhead between the controllers, and solve the model through linear programming. [4] proposed capacited controller placement model, taking the load into account, and optimize control link transmission delay under the premise to ensure the controller load balancing. [6] focusing on SDN flexible control problems, proposed heuristic algorithm based on k-center, which achieve a rapid deployment of dynamic computing controller.[7] analyzed the influence of the control node failure on network performance by comparing latency and load before and after the fault in the network, and proposed a controller deployment method based on Pareto optimization.

The difference between the methods above is confined to discuss several different indicators related to network service quality. They use the same kind of problem-solving ideas essentially that establish contact between the controller and the switch, changing the real problem into an integer programming problem firstly, solve the NP-hard problem by brute-force method or heuristic algorithm secondly and partition the same controller nodes to the same sub-domain finally. This idea merges the two steps calculation of sub-domains partition and controller deployment, without considering domains partition problem separately. For the actual network, the topological structure generally has community. So we can take advantage of community discovery algorithm to divide the network into a number of sub-domains firstly, making the node within the same sub-domains interconnect closely, and the node between the different sub-domains interconnect sparsely[8]. With ensuring control link connectivity between the controller and the switch, it separates the calculation process of domain partition and controllers placement, as well as reduces calculation complexity of the model.

This paper proposes a Capacitated Label Propagation Algorithm(CLPA) for large scale SDN domain partition of two steps based on Label Propagation Algorithm(LPA) [9].Step one: On the basis of improved LPA, it limits the spread of the maximum number of label hops and maximum difference of nodes number between domains to divide the network into domains that meets the load balancing and reliability requirements. Step two: According to average control link transmission delay, controllers were deployed in each sub-domain separately. Through transforming the multi-controller deployment problem into a multi-single-controller

978-1-4673-9904-3/16/$31.00 ©2016 IEEE

deployment issue, it makes the model complexity linearly scale with the network size.

Based on the simulation on Internet2 and topology zoo, it can be found that load balancing of the controller has been significantly improved compared with original LPA. Model computational complexity and average control link transmission delay have been greatly reduced, and reliability of the control link has been strengthened.

The problem model is defined in section II of this paper, and solutions are proposed in section III. Finally, the simulation experiments and computational complexity are discussed in Section IV.

II. PROBLEM FORMULATION

The network is modeled as an undirected graph $G=(V,E)$. $V=(v_1,v_2,\cdots,v_n)$ is the set of undirected graph nodes. $n=|V|$ represents the number of nodes in the network. E is the set of edges. $m=|E|$ represents the number of edges in the network. The network is divided into p domains as $C=\left(C_1,C_2,\cdots,C_p\right)$. Each domain contains any k $(1\le k\le n)$ nodes and a controller s_i, which can be presented as $C_i=(v_{t_1},v_{t_2},\cdots,v_{t_k},s_i)$. The set of network controllers is $S=(s_1,s_2,\cdots,s_p)$. All domains are not overlapped, and any node in the network belong to and only belong to one domain:

$$\forall i,j \quad i\ne j \quad C_i\cap C_j=\varnothing \tag{1}$$

$$\left|\bigcup_i C_i\right|=n \tag{2}$$

In order to describe and model the problem as well as the related algorithms, the following definitions are proposed:

Definition 1. Control link.

The link that transmits control information between the controller and the switch is called control link. There are two ways of implementation: the first is in-band transmission, using data plane traffic link to forward control information; the second is out-band transmission, using a dedicated link to forward control information. For large-scale networks, the number of switches and distance between switches are large, so the control link described in this paper uses in-band transmission.

Definition 2. Node labels $L(V)$.

Each network node is assigned to a different node label $L^0(V)=\left(L^0(v_1),L^0(v_2),\cdots,L^0(v_n)\right)$ when initialed, and $L^0(v_1)=l_1$, $L^0(v_2)=l_2$, $\cdots$, $L^0(v_n)=l_n$. After propagation q times, the nodes labels are $L^q(V)=\left(L^q(v_1),L^q(v_2),\cdots,L^q(v_n)\right)$, and $L^q(v_1)=l_{q_1}$, $L^q(v_2)=l_{q_2}$, $\cdots$, $L^q(v_n)=l_{q_n}$, $l_{q_k}\in\{l_1,l_2,\cdots,l_n\}$, $1\le k\le n$.

Definition 3. Connection state matrix $X=\left[x_{i,j}\right]_{n\times n}$.

It indicates the connection relationship between the controller and the switch in network. i, j denote switches and controllers node numbers. Matrix value represents the connection status. When $x_{i,j}=1$, it means that the switch v_i is controlled by the controller s_j.

$$x_{i,j}=\begin{cases}1 \text{ if switch i is assigned to controller j}\\ 0 \text{ if otherwise}\end{cases} \tag{3}$$

Definition 4. Shortest path length between adjacent nodes $d(v_i,v_j)$.

For large-scale network, the ground can not be regarded as plane. Suppose the earth is a sphere standard. Radius is r. The longitude and latitude of node v_i is α_i and β_i. The longitude and latitude of node v_j is α_j and β_j. When calculate longitude, east longitude is positive and west longitude is negative. When calculating latitude, south latitude is 90°+latitude value, and northth latitude is 90°-latitude value. The shortest path length between two points adjacent $\left(v_i,v_j\right)$:

$$d\left(v_i,v_j\right)=r\cdot\arccos\left(\sin\left(\beta_i\right)\sin\left(\beta_j\right)\cos\left(\alpha_i-\alpha_j\right)+\cos\left(\beta_i\right)\cos\left(\beta_j\right)\right) \tag{4}$$

Definition 5. Average control link transmission delay L_{avg}.

Since the distance between nodes is far, transmission distance will be the main factors affecting the signal propagation delay. Therefore, the signal propagation delay is represented by the distance between the node. There is only one controller in each domain, controlling all the switches. $D(v_i,s_j)$ is the Shortest path length between switch $v_i\in V$ and controller $s_j\in S$. For the controller set and the switch set, Average control link transmission delay L_{avg} is the average control link transmission delay between all switches to the respective controllers:

$$D\left(v_i,s_j\right)=\min_{v_1,\cdots,v_k}\left(d\left(v_i,v_1\right)+d\left(v_1,v_2\right)+\cdots+d\left(v_{k-1},v_k\right)+d\left(v_k,s_j\right)\right) \tag{5}$$

$$L_{avg}\left(V,S\right)=\frac{1}{n}\sum_{v_i\in V}\sum_{s_j\in S}D(v_i,s_j)\cdot x_{i,j} \tag{6}$$

Definition 6. Load balancing index B.

Controllers handle the messages submitted by switch. The larger the message volume is, the greater the controller load is, and performance of the controller will be affected. So the best way is all controllers having balanced load. Avoid some controllers being overload, and some controllers being idle. Divide the network into p domains, C_1, C_2, $\cdots$, C_p, and in each of them is placed one controller. Assuming that the average amount of information the switch v_k submit to the controller is b_k. Controller load balancing status can be

leveraged by the difference of message volume the switch submit among the domains:

$$B(C)=\max_{i}\sum_{v_p\in C_i}b_p-\min_{j}\sum_{v_q\in C_j}b_q \tag{7}$$

Definition 7. Reliability index R .

Suppose the network switching node has the same probability of failure, and each of the node is independent. So the more the number of node hops between the controller and the switch, the greater the probability of a communication link failure occurs. Reliability of the control links between nodes can be measured by the reciprocal average number of hops:

$$\begin{aligned}k(v_i,s_j)=\Big\{k\mid \min_{v_1,\cdots,v_k}\big(&d(v_i,v_1)+d(v_1,v_2)+\cdots\\&+d(v_{k-1},v_k)+d(v_k,s_j)\big),v_i\in V,s_j\in S\Big\}\end{aligned} \tag{8}$$

$$K_{s_j}^{v_i}=k(v_i,s_j)+1 \tag{9}$$

$$R=\frac{n}{\sum_{v_i\in V}\sum_{s_j\in S}K_{s_j}^{v_i}\cdot x_{i,j}} \tag{10}$$

For large-scale SDN, the model of optimizing control link propagation delay, reliability and controller load balancing is as follows:

$$\min\ L_{avg} \tag{11}$$

$$s.t.\quad B\le B' \tag{12}$$

$$R\ge R' \tag{13}$$

$$\sum_{j}x_{i,j}=1;\ \forall i \tag{14}$$

$$x_{i,j}\in\{0,1\};\ \forall i,j \tag{15}$$

Equation (11) is the target of optimizing, minimizing the average control link transmission delay. Equation (11) are constraints. B' in equation (12) is the limit of controller load balancing index, maintaining load balancing within a certain range. R' in equation (13) is the minimum control link reliability, which avoid excessive control link hops, enhancing links reliability. Equation (14) ensures that all network switches are assigned to only one controller. Equation (15) constrains the value of the connection state matrix, ensuring that there are only '1' and '0', which means connecting or not.

Above are descriptions of the model problem. In order to solve this integer programming problem, this paper proposes a two-step algorithm, and explain in detail in the next section.

III. ALGORITHM

[9] proposed a complex network community discovery algorithm based on label propagation, with no need of priori information, and objective function of optimization. According to the characteristics of the internal network structure, the network is divided into multiple domains with less computational and storage resources in linear time. However, due to uneven network structure, the difference in size between the domains is large when using LPA, resulting in uneven controller load. [5] proposed CPP deployment strategy based on K-center problem. Optimization target is to make the maximum length of the shortest link between switches and their respective controllers minimum.

This paper presents DPCS algorithm with two steps based on the above two algorithms. The first step of the algorithm is using an improved LPA algorithm for network partition. The size of the domain is limited to ensure the controller load balancing. Firstly, algorithm initializes node labels in the network(line 1). Then, each node begin to spread labels outwards and select the highest number of its neighbor nodes label as its new label(line 12). Finally, assign the nodes with same label to the same domain. After finite cycles, if the network status labels converge, which means consecutively updated label consistent, the program breaks the cycle(line 14), and save the last update as a result of the final results of partition. Besides, the maximum number of label propagation hops is constrained, stopping label propagating after certain hops, which limits domains radius and ensure reliability of control links. Before each update cycle, the sum of the average message volume in each domain is checked(line 8). If load balancing index B reaches the upper limit, the propagation of the maximum domain is stopped, which guarantees controller load balancing.

Step one：network domain partition

Input: Network topology $G=(V,E)$

Adjacent nodes matrix $A=\left[a_{i,j}\right]_{n\times n}$

Load balancing index upper limit B'

Reliability index lower limit R'

Output: Topology partition result C

1 $L^0(V)=\left(L^0(v_1),L^0(v_2),\cdots,L^0(v_n)\right)$

2 **for** $t=0;t=t+1$

3 $count(L^t(V))=\cup_{C_i}count\left(L^t(C_i)\right)$

4 $V_k=\varnothing$

5 **for each** $v_k\in V$ **do**

6 $A,v_k\rightarrow V_{adj}$

7 **for each** $v_i\in V_{adj}$ **do**

8 **if** $count\left(L^t(v_i)\right)-\min count\left(L^t(V)\right)\le B'$ && $\frac{1}{K_{v_i}}\ge R'$

9 $V_k=V_k\cup v_i$

10 **end**

11 **end**

12 $L^{t+1}(v_k)=\left\{l_i\middle|\max_{l_i}\left(countif\left(L^t(V_k)=l_i\right)\right),1\le i\le n\right\}$

13 **end**

14 **if** $L^{t+1}(V)=L^t(V)$

15 $C=\left(C_1,C_2,\cdots,C_p\right)$

16 break

17 **end**

18 **end**

The second step of the algorithm is selecting positions to deploy the controllers. Since controller load balancing and control link reliability have been taken into account when partitioning domains, only the link delay needs to be considered when choosing controller positions. Search the best positions through the entire domain to minimize average control link delay between switches and controllers(line2). Finally, update the connection state matrix(line 5).

Step two：controller placement

Input: Topology partition result C

Output: Connection state matrix $X=\left[x_{i,j}\right]_{n\times n}$

1 **for each** $C_i \in C$ **do**

2 $s_j=\left\{v_k \middle| \min_{v_k} L_{avg}\left(C_i,v_k\right), v_k \in C_i\right\}$

3 $S=S\cup s_j$

4 **for each** $v_i \in C_i$ **do**

5 $x_{i,j}=1$

6 **end**

7 **end**

8 $x_{i,j} \to X$

IV. SIMULATION

A. Experimental Environment

1. This paper select Internet2 Network Advanced Layer 2 Service as an experiment topology, and select several topologies from topology zoo to further verify algorithms. Internet2, established in 1996, is jointly supported as the next generation of the Internet by more than 120 universities, research institutes and companies in the United States. Topology zoo project is led by the University of Adelaide and supported by the Australian Government, which has collected more than 250 network topologies from all over the world with high reference and research value.

2. Hardware environment of experiment is PC with Intel Core i7-3770 CPU 3.40 GHz, RAM 4GB. CLPA algorithm and original LPA are simulated on MATLAB 2013. Linear programming in CCP is solved by CPLEX 7.0.

B. Parameter Settings

1. In order to simplify the experiment, following assumptions are made: without considering the cable bending in practice, the shortest distance between two points is represented by a straight line on the sphere; without considering the difference of controllers, all controllers have the same capacity, handling ability and transmission speed. without considering the difference of switches, all switches have the same message volume, which can be set according to specific situation in practice.

2. Parameter B' can be set manually in CLPA and CCP, so there is no need to compare the index B in experiment. Therefore, set $B'=2$, and compare average delay and control link reliability under the premise to ensure the controller load balancing.

C. Experimental Analysis

1) Algorithm Complexity

For NP-hard problem, the best results can only be got by the brute-force method. When deploying p controllers in the network with n nodes, the complexity for brute-force method is $O\left(n^p \cdot p^n\right)$. Assume that $n=50$ and $p=5$, the complexity is 10^{34}, which is far beyond the current computing power.

[13]proposed CCP algorithm, solving this integer linear programming model By a two-step algorithm. Assume that there are n nodes and m edges in the network. Firstly, the linear relaxation problem is solved by dichotomy in the first phase and then the lower limit of maximum distance between the controller and the switch r is calculated. The complexity is $O\left(n^{3.5}L^2 \lg m\right)$ if using interior point method to solve the linear program. L is the number of linear programming constraints. The second phase modifies r on the basis of the first phase, and calculate the suboptimal solutions, whose complexity is $O\left(n \lg n\right)$. Therefore, the total complexity is $O\left(n^{3.5}L^2 \lg m\right)$.

For CLPA algorithm, complexity linearly scale with the network size. In the network with n nodes and m edges, initialing all nodes label needs $O\left(n\right)$.In each cycle, calculating number of nodes in each domain needs $O\left(n\right)$. Looking up the table to find the propagation hops of each label needs $O\left(n\right)$. Counting the label number of adjacent nodes needs $O\left(m\right)$. Selecting the maximum label number of adjacent nodes as the new label needs $O\left(m\right)$. In summary, the total complexity of the algorithm is $O\left(n+m\right)$, which is linear with the scale of the network $\left(n,m\right)$.

TABLE I. ALGORITHM COMPLEXITY

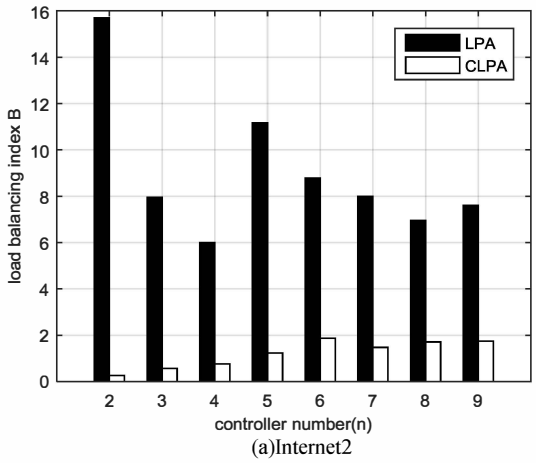
algorithm	complexity
Brute-force	$O\left(n^p \cdot p^n\right)$
CCP	$O\left(n^{3.5}L^2 \lg m\right)$
CLPA	$O\left(n+m\right)$

2) Analysis of simulation results

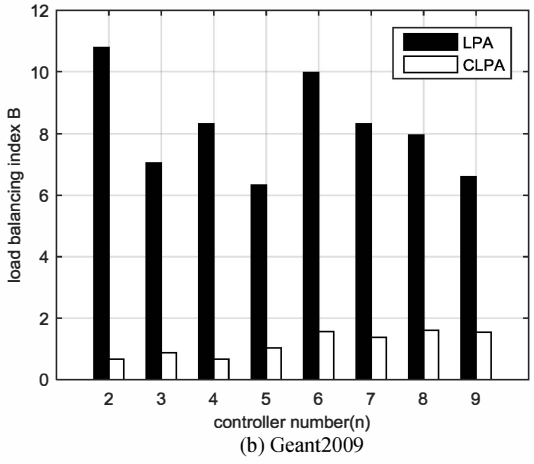
In this section, original LPA, CCP and CLPA are simulated on Internet2 and topology zoo. And then, we analyze several indexes of average control link delay, the controller load balancing, and link reliability.

Firstly, compare load balancing index of original LPA and CLPA on Internet2, Geant2009 and Canerie topology separately. The result is as figure 1. Load balancing index of

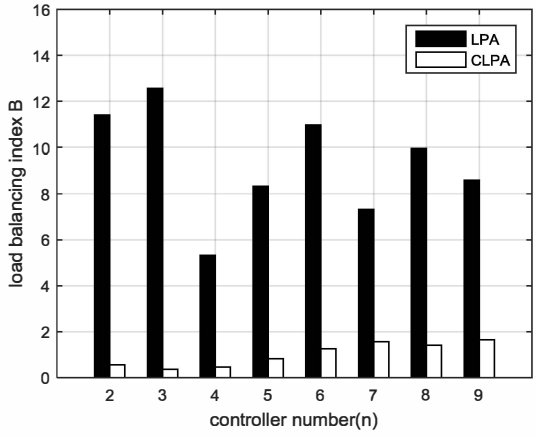
the original LPA can't be controlled, and fluctuates greatly for different controller numbers. And it is greatly higher than CLPA, which will cause uneven load distribution. This is because original LPA doesn't limit the size of domains, and the partition result depends on the structure of the network, without considering load balancing. CLPA limits the size of domains. When the switch number of a domain is excessive, the label propagation of the domain will be stopped from the next cycle. Results show that the controller load balancing index is stable and low using CLPA, which achieve controller load balancing effectively.



(a)Internet2



(b) Geant2009



(c) Canerie

Figure 1. Compare LPA and CLPA in different topologies

Then, compare average control link delay and reliability of CLPA and CCP. The result is as figure 2. Average control link delay L_{avg} of two algorithms declines as a downward convex curve. The same as figure 3, reliability index R of two algorithms rises as an upward convex curve.

Through comparing, L_{avg} and R of CLPA are better than CCP under the same number of controllers. Curve of CLPA advantaging over CCP first rises and then falls. The advantage curve rises because CCP algorithm is based on constrained packing problem with a maximum controlling range ε. When the nodes distribution of the topology is uneven, ε is too large for the concentrated part, which means that the restraint is too relaxed. However, label propagation will not be affected by the uneven distribution. Then the advantage curve declines because the domains structure is too simple when the number of controllers rises. There is limited space for further optimizing.

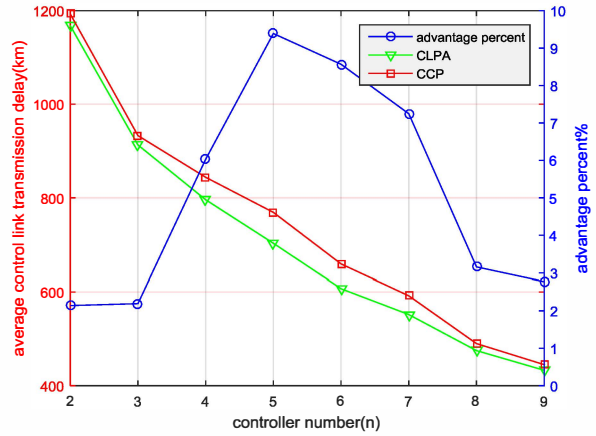


Figure 2. Control link transmission delay for different controller numbers

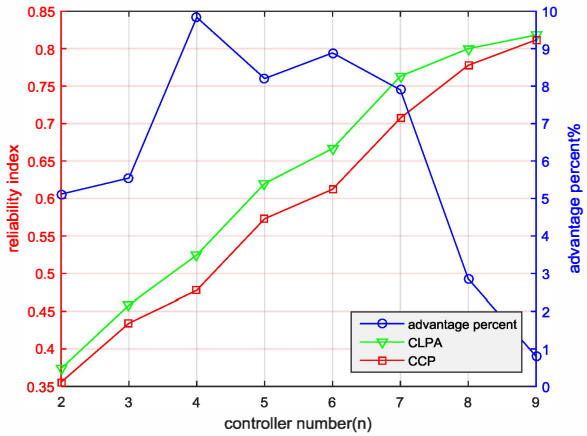


Figure 3. Reliability index for differernt controller numbers

Finally, we select 10 topologies from topology zoo to further verify algorithms. $\left\lfloor \frac{n}{15} \right\rfloor \sim \left\lfloor \frac{n}{5} \right\rfloor$ (n is the number of nodes)controllers are deployed by CLPA and CCP separately in each topology. Then we calculate average advantage CLPA over CCP of the average delay and reliability index. The result is as figure 4. CLPA algorithm performance is superior than CCP Algorithm in all 10 topologies, and the extent of optimizing depends on the topology structure. If the nodes distribution of the topology is uneven, like Geant2009, the advantage of CLPA will be greater.

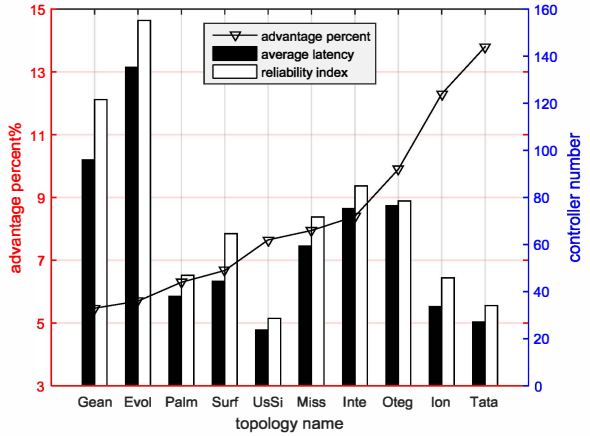


Figure 4. Advantage of CLPA in different topologies

V. CONCLUSION

For the SDN controller deployment problem of multi-controller architecture, this paper proposed CLPA consisting of two steps on the basis of LPA. Compared with the original LPA, CLPA modifies original LPA, and considers the load, as well as reliability, which will adapt to SDN better. Compared with CCP, CLPA reduces model complexity, and improves control link transmission delay and reliability index. Finally, the algorithm is tested on Internet2 and topology zoo.

In this paper, the number of switches and the number of hops are used to measure the controller load and link reliability which are slightly coarse-grained. There will be more detailed considerations in the next work.

REFERENCES

[1] ZUO Q Y, CHEN M, ZHAO G S. Research on OpenFlow-based SDN technologies [J]. Journal of Software, 2013, 24(5): 1078-1097.

[2] TOOTOONCHIAN A, GANJALI Y. HyperFlow: a distributed control plane for OpenFlow[C]//Proceedings of the 2010 internet network management conference on Research on enterprise networking. 2010: 3-3.

[3] HELLER B, SHERWOOD R, MCKEOWN N. The controller placement problem[C]//Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012: 7-12.

[4] SALLAHI A, ST-HILAIRE M. Optimal model for the controller placement problem in software defined networks[J]. Communications Letters, IEEE, 2015, 19(1): 30-33.

[5] YAO G, BI J, LI Y, et al. On the capacitated controller placement problem in software defined networks[J]. Communications Letters, IEEE, 2014, 18(8): 1339-1342.

[6] LANGE S, GEBERT S, SPOERHASE J, et al. Specialized heuristics for the controller placement problem in large scale SDN networks[C]// ITC 27: Teletraffic Congress, 2015 27th International. IEEE, 2015: 210-218.

[7] HOCK D, HARTMANN M, GEBERT S, et al. Pareto-optimal resilient controller placement in SDN-based core networks[C]// ITC : Teletraffic Congress, 2013 25th International. IEEE, 2013: 1-9.

[8] LUO Z G, DING F, JIANG X Z, et al. New progress on community detection in complex networks[J]. Journal of National University of Defense Technology, 2011, 33(1): 47-52

[9] RAGHAVAN U N, ALBERT R, KUMARA S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2007, 76(3 Pt 2):-.

[10] The Internet2 Community [EB/OL]. Internet2 Service. [2015-03-02]. http://www.internet2.edu /network/ose/.

[11] KNIGHT S, NGUYEN H X, FALKNER N, et al. The internet topology zoo [J]. Selected Areas in Communications, IEEE, 2011, 29(9): 1765-177