Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A *K* self-adaptive SDN controller placement for wide area networks*#

Peng XIAO[1,2], Zhi-yang LI[†‡1], Song GUO[3], Heng QI[4], Wen-yu QU[5,1], Hai-sheng YU[4]

([1]*School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*)

([2]*School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China*)

([3]*School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan*)

([4]*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*)

([5]*School of Computer Software, Tianjin University, Tianjin 300072, China*)

[†]E-mail: lizy0205@gmail.com

Received Oct. 21, 2015; Revision accepted Mar. 30, 2016; Crosschecked June 8, 2016

Abstract: As a novel architecture, software-defined networking (SDN) is viewed as the key technology of future networking. The core idea of SDN is to decouple the control plane and the data plane, enabling centralized, flexible, and programmable network control. Although local area networks like data center networks have benefited from SDN, it is still a problem to deploy SDN in wide area networks (WANs) or large-scale networks. Existing works show that multiple controllers are required in WANs with each covering one small SDN domain. However, the problems of SDN domain partition and controller placement should be further addressed. Therefore, we propose the spectral clustering based partition and placement algorithms, by which we can partition a large network into several small SDN domains efficiently and effectively. In our algorithms, the matrix perturbation theory and eigengap are used to discover the stability of SDN domains and decide the optimal number of SDN domains automatically. To evaluate our algorithms, we develop a new experimental framework with the Internet2 topology and other available WAN topologies. The results show the effectiveness of our algorithm for the SDN domain partition and controller placement problems.

Key words: Software-defined networking (SDN), Controller placement, *K* self-adaptive method
http://dx.doi.org/10.1631/FITEE.1500350 **CLC number:** TP393

1 Introduction

Historically, control plane functions in traditional networks have been tightly coupled to the data plane. The software-defined networking (SDN) concept (Kirkpatrick, 2013) has caused a paradigm shift in communication networks, which allows the separation of control and data planes, i.e., moving complex functions from devices in a network to sophisticated dedicated controller instances. The most popular example of SDN is OpenFlow (McKeown *et al.*, 2008), where an OpenFlow controller defines rules for switches on how to handle packets. Thus, the controller placement problems are becoming increasingly important.

For the local area network (LAN), the controller placement problem is simple. In general, only one

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61432002, 61370199, 61370198, 61300187, and 61402069), the Fundamental Research Funds for the Central Universities, China (Nos. DUT15QY20, DUT15TD29, and 3132016029), and the Prospective Research Project on Future Networks from Jiangsu Future Networks Innovation Institute, China

A preliminary version was presented at the IEEE/CIC International Conference on Communications in China, Shanghai, China, Oct. 13–15, 2014

ORCID: Zhi-yang LI, http://orcid.org/0000-0002-5396-3447

SDN controller is adequate to work in most LANs because a LAN is less affected by the propagation delay compared to a wide area network (WAN). However, a WAN is normally characterized by long propagation delay and scarce bandwidth. One of the most urgent challenges in deploying SDNs in WANs is the controller placement problem. For example, as shown in Fig. 1, where should controllers C_1 and C_2 be placed in a WAN and which controller, C_1 or C_2, should be selected by the OpenFlow switch S_1? These are still open questions and have attracted much attention recently.

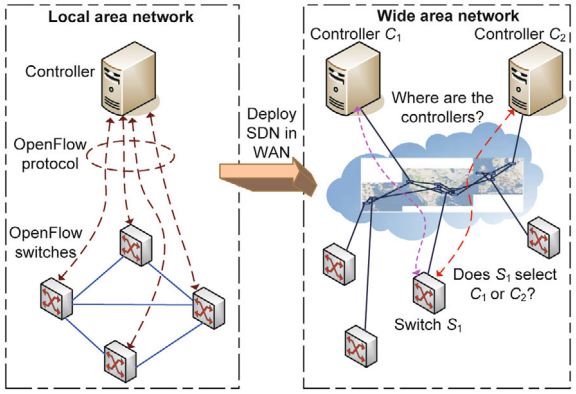

Fig. 1 The placement problem in a wide area network

A large-scale network is usually partitioned into several small ones due to numerous reasons, e.g., privacy, scalability, incremental deployment, and security (Xie *et al.*, 2012; Lin *et al.*, 2013). For SDN partitioning, a large network is likely to be divided into multiple SDN domains. Each SDN domain runs one controller, such as Floodlight (http://www.projectfloodlight.org/floodlight/), NOX (Gude *et al.*, 2008), and Beacon (https://openflow.stanford.edu/display/Beacon/Home/). An SDN domain can be a sub-network in a data center (DC), an enterprise network, or an autonomous system (AS). In this study, we consider the 'best' controller placement that minimizes propagation delays and improves reliability in a WAN partitioned into multiple AS domains.

The most relevant work can be found in Heller *et al.* (2012). The authors examined the impacts of placements on average-latency and worst-case latency on real topologies. However, they treated WAN as a whole rather than as multiple SDN domains and ignored the reliability of each controller. While propagation latency is certainly a significant design metric, we argue that reliability and load balancing design are also essential parts for operational SDNs. Heller *et al.* (2012) assumed that nodes are always assigned to their nearest controller using latency as the metric. In average-latency placement, the number of nodes per controller is imbalanced and ranges from 3 to 13 when the number of controllers is 4 (Fig. 2). The more nodes a controller has to control, the heavier the load on that controller will be. From Fig. 2, we can see the imbalance between controller 2 and controller 3. With regard to controller failure tolerance, Hock *et al.* (2013) optimized the placement of controllers, called Pareto-based optimal controller placement (POCO). However, their method causes the inter-controller broadcast storm and needs time to reassign nodes. Heller *et al.* (2012) and Hock *et al.* (2013) assumed that the mapping between a switch and a controller is configured dynamically, as in ElastiCon (Dixit *et al.*, 2013). The dynamic allocation can improve the scalability and reliability of the SDN deployed in a LAN, but it is not suitable for a WAN. Usually, the propagation latency is larger than the queuing delay in the network, and the dynamic mapping between a switch and a remote controller will significantly affect the response time of the WAN. Moreover, switch migrations are complex tasks with more overhead.

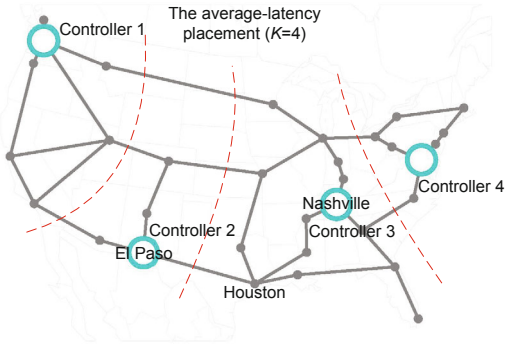

Fig. 2 Four partitions based on the average-latency placement

Motivated by these analyses, the SDN domain partition problem for a WAN has been studied (Xiao *et al.*, 2014). We first use spectral clustering to partition the WAN into several SDN domains, each with its own controller, similar to the domain name system (DNS). A single controller can be enough to manage a small network, and the backup controller

can reduce the impact of failure of a single controller. There are at least four reasons why we adopt the divide-and-conquer philosophy: (1) It facilitates load balancing and ensures reliability in the SDN infrastructure; (2) The partition of SDN domains can help reduce inter-controller broadcast storm, especially in large-scale WANs; (3) There are no latencies in reassigning nodes to their new controller because of static allocation; (4) It fits the layered model of a WAN and is easy for maintenance and expansion. In contrast, an over-complicated controller plane is hard to achieve and maintain.

Although the placement algorithm (Xiao *et al.*, 2014) may obtain the SDN domain partition results, the number of SDN domains (K) needs to be set manually. In this study, we focus on finding a K self-adaptive SDN controller placement for WAN and exploiting the structure of eigenvectors to determine automatically the number of SDN domains.

Compared with Xiao *et al.* (2014), the major contributions of this study are listed as follows:

1. We propose a K self-adaptive SDN controller placement for a WAN based on the matrix perturbation theory.

2. We propose an alternative approach which relies on the structure of eigenvectors to estimate the optimal number of SDN domains.

Experimental results show that our methods can solve the SDN controller placement problem and determine the number of SDN domains automatically.

2 Related work

Currently, there are mainly two categories of controllers in the SDN control plane: single controller and distributed controllers.

2.1 Single controller

Examples of the single controller include Floodlight (http://www.projectfloodlight.org/floodlight/), NOX (Gude *et al.*, 2008), Beacon (https://openflow.stanford.edu/display/Beacon/Home/), Maestro (Cai *et al.*, 2010), SNAC (http://groups.geni.net/geni/raw-attachment/wiki/GEC9Demo-Summary/), Trema (http://trema.github.io/trema/), etc. Floodlight is an enterprise-class, Apache-licensed, Java-based OpenFlow controller. It is supported by a community of developers, including a number of engineers from Big Switch Networks. NOX is a typical example of controller realization, aiming to simplify the management of switches in enterprise networks. Its constituent components, control granularity, switch abstraction, and basic operation are discussed in a NOX-based network. Beacon (Erickson, 2013) is a fast, cross-platform, modular, Java-based OpenFlow controller which supports both event-based and threaded operations. Shalimov *et al.* (2013) showed that Beacon is a pretty good controller. Cai *et al.* (2010) proposed Maestro, which keeps the simple programming model for programmers and exploits parallelism in every corner with additional throughput optimization techniques. These physically centralized control planes can be adapted for DCs but are not suitable for wide multi-technology multi-domain networks.

Recently, the concept of physically distributed SDN control plane has been proposed, including DISCO (Phemius *et al.*, 2014), Onix (Koponen *et al.*, 2010), HyperFlow (Tootoonchian and Ganjali, 2010), DIFANE (Yu *et al.*, 2010), Devolved (Tam *et al.*, 2011), and ElastiCon (Dixit *et al.*, 2013). Kreutz *et al.* (2015) found that most distributed controllers offer weak consistency semantics; i.e., data updates on distinct nodes will eventually be updated on all controller nodes. This implies that there is a period of time, in which distinct nodes may read different values (old value or new value) for the same property. On the other hand, the controller will take more time to communicate with other controllers and switch in WANs with long propagation delay, aggravating the system performance significantly.

2.2 Distributed controllers

Two key problems in SDNs with distributed controllers are: (1) how to obtain a global view of the entire network at each controller so as to maintain a consistent network state, and (2) how to deploy an optimal number of controllers such that the best performance can be achieved.

To address the first problem, Yin *et al.* (2012) have proposed the inter-SDN (SDNi) domain protocol, which acts as an interface mechanism to coordinate the behaviors of SDN controllers in the SDN domains. However, SDNi still lacks a semantic network model and an ontology-based model to ensure the extensibility of its transport mechanisms and syntax.

Lin *et al.* (2013) have proposed the east-west bridge solution to enable different controllers from different vendors to work together. They have deployed this solution with two use cases to four SDN networks, such as Internet2 in the USA and CERNET in China. In this study, we focus on the controller placement problem in WANs. We assume that the first problem has been solved perfectly, and the controller working in each SDN domain can exchange information.

The second problem is about controller placement. Heller *et al.* (2012) over-simplified the problem by modeling it as a facility or a warehouse location problem, in which only the latency of transmission from nodes to their controllers was considered and the WAN topology was treated as a whole rather than as multiple SDN domains. These lead to heavy load or failure at some controllers near the switches with intensive traffic. In view of the characteristics of the traditional WAN, a divide-and-conquer philosophy is desired for the deployment of SDNs in WANs. A large WAN is always partitioned into several small SDN domains to ensure stability, privacy, management, security, and so on. Therefore, it is necessary to develop a method to address these challenges for the SDN controller placement problem in a WAN. In this study, we focus on using a K self-adaptive SDN controller placement to partition a WAN topology into several small SDN domains, as well as on placing controllers to achieve low latency and high reliability in each SDN domain.

3 Problem description and system model

In this section, we briefly introduce the definition of SDN domain partition for a WAN and discuss the optimization placement metrics we intend to study.

3.1 Problem description

WAN is a network that covers a broad area, including many regions or countries. In the SDN, the controller acts as an information collector and operator for its managed switches. In this regard, the response time between the switch and controller significantly affects the performance of the SDN. Furthermore, the response time of the controller is determined by the propagation delay and the controller's load. For example, as shown in Fig. 2, the propagation delay between Houston and Nashville is about 5.01 ms, and the time delay between Houston and El Paso is about 5.44 ms. In average-latency placement, Heller *et al.* (2012) considered only the propagation delay, so the switches in Houston were assigned to the third controller deployed in Nashville but not the second controller deployed in El Paso. In general, the queuing delay of a network is much longer than the propagation latency. From Fig. 2, we can see the imbalance between the second and third controllers. When the third controller is overloaded, the queuing delay in the network is longer than the propagation latency and is rising steadily. Therefore, Heller *et al.* (2012) simply assigned switches to their closest controller, which may lead to controller overload and instability.

As a new deployment in a WAN, controller placement influences every aspect of the SDN network in WAN, from performance to security. In this study, we narrow our focus to two essential factors, balanced partition and propagation latency.

1. Balanced partition

Load balancing and reliability are two important indicators of controller performance. Tootoonchian *et al.* (2012) focused on controller performance and found the limitations of a controller's service ability. With enough delay and overload of the controller, real-time tasks become infeasible, while others may slow down unacceptably. By partitioning the WAN into several small balanced SDN domains, the service ability of a controller with fewer and balanced nodes will be improved greatly, and the inter-controller broadcast storm will be reduced sharply, which will greatly reduce the queuing delay.

2. Propagation latency

After considering the reliability of partition, network latency is certainly a significant design metric in long-propagation-delay WAN. Network latency includes four parts: propagation latency, processing latency, queuing latency, and transmission latency. For WANs, the propagation latency is longer than the other latencies, and the effect of the other latencies is so small that it can be ignored. Regardless of the exact form, in the case of WAN, the propagation delay affects the controller's ability to respond to network events. Based on Heller *et al.* (2012), we

also narrow our focus to propagation latency and select it as a significant design metric. We assume that propagation latency is the response time of the controllers, and the 'best' placement must ensure that the latency of each SDN domain is the minimum.

We need to find a placement solution to balance the load and reduce the latency. In the next subsection, the quantitative analysis of our placement with a global optimization goal will be proposed.

3.2 System model

We model the network as a graph, $G(S, E)$. The node set S represents the nodes in the network topology, i.e., the OpenFlow switches deployed to the different cities, and the edge set E represents the network links between the cities. We partition G into K subgraphs, namely, SDN domains N_i $(i = 1, 2, \ldots, K)$.

Definition 1 If we partition G into K subgraphs, namely, N_i $(i = 1, 2, \ldots, K)$, then N_i can be defined as $N_i(S_i, E_i)$. Clustering the nodes in S is equivalent to partitioning the set of vertices S into mutually disjoint subsets $S_1,\ S_2,\ \ldots,\ S_K$ according to some similarity measure, namely,

$$S = \bigcap_{i=1}^{K} S_i, S_i \cap S_j = \oslash,\ i \neq j,\ i, j = 1,\ 2,\ \ldots,\ K, \tag{1}$$

where K is the number of SDN domains and S_i denotes the ith SDN domain. The nodes in S are ordered according to the cluster they are in:

$$\underbrace{\{s_1, s_2, ..., s_{i_1}\}}_{n_1} \in S_1,\ \underbrace{\{s_{i_1+1}, s_{i_1+2}, ..., s_{i_2}\}}_{n_2} \in S_2,$$
$$\ldots,\ \underbrace{\{s_{i_{K-1}+1}, s_{i_{K-1}+2}, \ldots, s_{i_K}\}}_{n_K} \in S_K. \tag{2}$$

We want to find a partition of the SDN domains such that the edges in different clusters have a very low weight (which means that OpenFlow switches in different clusters are dissimilar from each other) and the edges within a cluster have a high weight (which means that OpenFlow switches within the same cluster are similar to each other). Furthermore, controller must be placed in the clustering center to ensure the maximum performance of the sub-network. Obviously, we want many edges within clusters and few edges between clusters. In addition to the minimum cut requirement, we require that the partition be as balanced as possible. This is a typical data clustering problem using a graph model. Inspired by previous work on spectral clustering (Shi and Malik, 2000; Wauthier *et al.*, 2012; Mall *et al.*, 2013; Liu *et al.*, 2014), we propose our methods to solve the SDN partition problem, which can provide balanced partitions and average the load of each controller.

The weight on each link, w_{ij}, is a function of the similarity between switches s_i and s_j. The weighted adjacency matrix of the graph is $\boldsymbol{W} = (w_{ij})_{i,j=1,2,\ldots,n}$. Inspired by the '$N_{\text{cut}}$' proposed by Shi and Malik (2000), we can obtain some balanced SDN domains that minimize similarity between sets and maximize similarity within a set, satisfying the following partition objective function:

$$\text{SDN}_{\text{cut}} = \sum_{i=1}^{K} \frac{\sum\limits_{x \in N_i, y \in G - N_i} w_{xy}}{\sum\limits_{x \in N_i, y \in G} w_{xy}}. \tag{3}$$

This objective function favors balanced SDN domains and minimizes the number of domain edges, which results in balanced switches and links in each SDN domain.

After solving the WAN partition problem, we now introduce the controller placement problem in each sub-network. Each SDN domain has only one master controller. Where should the controller be located to ensure the performance of a single SDN domain? It is called a facility location problem and occurs in much context (Heller *et al.*, 2012). Let our placement model be

$$\min \sum_{c_i \in C} \sum_{s \in N_i} \text{dist}(s, c_i), \tag{4}$$

where C is a given placement solution and $\text{dist}(s, c_i)$ represents the shortest path from node $s \in N_i$ to node $c_i \in C$.

The key idea behind is to first identify the partitions with balanced cuts, and then assign the controller location to the center of each partition, which has the shortest paths to all switches in the same SDN partition. Clearly, we can use the function to find the 'best' placement solution C from the set of all possible placements, along with the minimum objective, which can balance the load and reduce the latency.

In the following section, we introduce an approximation algorithm to optimize the problem by using spectral clustering.

4 Controller placement algorithm

Our approach for SDN controller placement is based on concepts from the spectral graph theory. The core idea is to use matrix theory and linear algebra to study the properties of the similarity matrix $\boldsymbol{W}$ and the Laplacian matrix. All the related theories and the idea of using eigenvectors of the Laplacian for finding partitions of graphs can be traced in Shi and Malik (2000), Wauthier *et al.* (2012), Mall *et al.* (2013), and Liu *et al.* (2014). In this section, we first introduce our methods for building the similarity matrix $\boldsymbol{W}$ and the Laplacian matrix $\boldsymbol{L}$. This is the first and the most important step of the spectral clustering algorithm. Then the K self-adaptive method is proposed to decide the optimal number of SDN domains automatically, which can help achieve the partition objective. Lastly, we describe the whole placement algorithm based on the spectral theory. To achieve the placement objective, we use the k-means method to cluster the nodes and select the center of each domain as the controller location.

4.1 Similarity function

In recent years, spectral clustering has become one of the most popular modern clustering algorithms, and it has been applied in machine learning, text summarization, social networks, etc. The success of such algorithms depends heavily on the choice of the similarity matrix $\boldsymbol{W}$. From the analysis of the propagation delay of WAN topology, we tend to select the propagation delay as the weight of the similarity matrix.

In the WAN topology G, the switches s_1, s_2, ..., s_n can be deployed in the nodes of the WAN topology, and their similarities w_{ij} can be measured according to the similarity function (which is symmetric and non-negative):

$$\xi = \sin^2(\alpha/2) + \cos(\mathrm{lat}_i)\cos(\mathrm{lat}_j)\sin^2(\beta/2), \quad (5)$$

$$w_{ij} = \frac{2\arcsin\sqrt{\xi \times 6378.137}}{V_c}, \quad (6)$$

where $s_i(\mathrm{lat}_i, \mathrm{lon}_i)$ and $s_j(\mathrm{lat}_j, \mathrm{lon}_j)$ represent the latitude and longitude of points s_i and s_j, respectively, $\alpha = |\mathrm{lat}_i - \mathrm{lat}_j|$, $\beta = |\mathrm{lon}_i - \mathrm{lon}_j|$, V_c is the speed of light propagation in optical fibers, and the radius of the Earth is 6378.137 km. We denote the corresponding similarity matrix by $\boldsymbol{W} = (w_{ij})_{i,j=1,2,\ldots,n}$, which can be used to evaluate the propagation latencies between the nodes.

Finally, the Laplacian matrix $\boldsymbol{L} = [L_{ij}]$ for the SDN domain partition is defined, where

$$L_{ij} = \begin{cases} -w_{ij}, & i \neq j, \\ \sum_{k=1}^{n} w_{ik}, & j = i, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

In SDN partitioning, the spectral decomposition of $\boldsymbol{L}$ can be used to approximately minimize $\mathrm{SDN}_{\mathrm{cut}}$, which tries to achieve balanced SDN domains in terms of the size.

4.2 K self-adaptive method

Although spectral clustering has many advantages and impressive performances, one of the common shortcomings is that the cluster number must be decided in advance. Some scholars have proposed different adaptive spectral clustering algorithms (Zelnik-Manor and Perona, 2004; Wang *et al.*, 2007). From the overview of their analyses, every data point can be regarded as an attribute sequence made up of all its attribute values. In this way, the similarity between any two points can be measured by the balanced closeness degree of the attribute sequences. Since the calculation of the balanced closeness degree does not need extra parameters, the impact of the parameters is eliminated. However, the methods in Zelnik-Manor and Perona (2004) and Wang *et al.* (2007) have higher cost and time complexities. In this section, we propose an approach that relies on the structure of the eigenvectors to automatically determine the optimal number of SDN domains. Based on the matrix perturbation theory (Bach and Jordan, 2003; Tian *et al.*, 2007; von Luxburg, 2007; Rebagliati and Verri, 2011) and k-way partition (Ng *et al.*, 2001), the difference between the kth and $(k+1)$th eigenvalues is called 'eigengap', which can be used directly to perform clustering.

Suppose the similarity matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq \ldots \geq \lambda_n$ be its eigenvalues, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_n$ the associated eigenvectors. For simplicity, we would like to call $\lambda_1 \geq$

$\lambda_2 \geq \ldots \geq \lambda_k$ the first k largest eigenvalues of $\boldsymbol{W}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ the first k largest eigenvectors of $\boldsymbol{W}$. Matrix $\boldsymbol{W}$ can be decomposed into the following form:

$$\boldsymbol{W} = \boldsymbol{X} \wedge \boldsymbol{X}^{\mathrm{T}}, \tag{8}$$

where $\wedge = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix with nonnegative singular eigenvalues in descending order along the diagonal, that is, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$; $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ is a matrix formed by stacking the eigenvectors of $\boldsymbol{W}$ in columns.

Let $\boldsymbol{M}$ be a matrix in subspace r which is spanned by the columns of $\boldsymbol{X}$, i.e., $\boldsymbol{M} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_r)$. The vectors $\boldsymbol{M}_i$ $(i = 1, 2, \ldots, n)$ can be defined as the rows of the truncated matrix $\boldsymbol{M}$, as follows:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \\ \vdots \\ \boldsymbol{M}_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1r} \\ x_{21} & x_{22} & \ldots & x_{2r} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nr} \end{pmatrix}, \tag{9}$$

$$\sigma = \sqrt{\sum_j M_{ij}^2}, \quad P_{ij} = \frac{M_{ij}}{\sigma}. \tag{10}$$

We construct a matrix from $\boldsymbol{M}$ by reforming each of $\boldsymbol{M}$'s rows to have unit length, such as $P_{ij} = M_{ij}/\sigma$ mentioned in Eq. (10). Under the above conditions, we can obtain the following result (Tian *et al.*, 2007):

Theorem 1 Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of matrix $\boldsymbol{W}$, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ be the first k eigenvectors of $\boldsymbol{W}$ satisfying Eq. (8), respectively. Let $\boldsymbol{M} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$. Form the matrix $\boldsymbol{P}$ from $\boldsymbol{M}$ by reforming each of $\boldsymbol{M}$'s rows to have unit length and let $\boldsymbol{P} = [\boldsymbol{p}_1^{\mathrm{T}}, \boldsymbol{p}_2^{\mathrm{T}}, \ldots, \boldsymbol{p}_n^{\mathrm{T}}]$, where $\boldsymbol{p}_i$ is the ith row vector of $\boldsymbol{P}$. Then

$$\cos\theta_{ij} = \frac{|\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{p}_j|}{\|\boldsymbol{p}_i\| \cdot \|\boldsymbol{p}_j\|} = \begin{cases} 1, & v_i \text{ and } v_j \text{ belong to a domain,} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

So, we can see that the decomposition of $\boldsymbol{W}$ can help obtain the clusterings. After obtaining $\wedge$, we can calculate the eigengap vectors as follows:

$$\{g_1, g_2, ..., g_{n-1} |\ g_i = \lambda_i - \lambda_{i+1}\},\ i = 1, 2, \ldots, n-1. \tag{12}$$

We can compute the suitable number of SDN domains K by analyzing the eigengap vectors as follows, derived from the matrix perturbation theory mentioned above:

$$K = \arg\max_i \{g_i\}. \tag{13}$$

Based on Eq. (13), the number of SDN domains can be determined by the associated eigengap values. Given a network topology, we can infer automatically the suitable number of SDN domains by exploiting the structure of the eigenvectors.

4.3 Spectral clustering placement algorithm

Now we would like to state our self-adaptive spectral clustering algorithm for the SDN controller placement problem in WAN. The whole algorithm is outlined in Algorithm 1.

Algorithm 1 Self-adaptive spectral clustering

1: **Input**: Network topology, G.
2: **Output**: Clusters $N_1, N_2, \ldots, N_K$ with $N_i = \{j | \boldsymbol{y}_j \in C_i\}$, and the locations of the controllers.
3: Construct similarity matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ according to Eq. (6).
4: Compute $\wedge$ according to Eq. (8).
5: Compute eigengap vectors according to Eq. (12).
6: Obtain the optimal number of SDN domains K according to Eq. (13).
7: Compute the graph Laplacian matrix $\boldsymbol{L}$.
8: Compute the first k eigenvectors of $\boldsymbol{L}$, i.e., $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, ..., $\boldsymbol{v}_k$.
9: Let $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ as columns.
10: **for** $i = 1$ to n **do**
11: Let $\boldsymbol{y}_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of $\boldsymbol{V}$.
12: **end for**
13: Cluster the points $(\boldsymbol{y}_i)_{i=1,2,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, C_2, \ldots, C_K$.
14: Return SDN domains $N_1, N_2, \ldots, N_K$ and the center of each domain.

As shown in Algorithm 1, the similarity matrix $\boldsymbol{W}$ is constructed by Eq. (6) (line 3). Then we use the eigengap to discover the clustering stability and decide the 'best' partition number K automatically (lines 4–6). After obtaining the optimal number of SDN domains (K), we can calculate the Laplacian matrix $\boldsymbol{L}$ and the first k eigenvectors of $\boldsymbol{L}$ (lines 7–8). Next, we construct a new sub-vector $\boldsymbol{V}$

corresponding to the first k eigenvectors (lines 9–12). Finally, we use the k-means algorithm to cluster the points into each partition and obtain the center of each partition (lines 13–14). The k-means algorithm can achieve a good placement metric (Eq. (4)). Our self-adaptive spectral clustering algorithm does not need to pre-specify the number of SDN domains, and it can obtain automatically the optimal number of SDN domains by calculating the eigengap, which are proved by the following experiments.

In Algorithm 1, solving a standard eigenvalue problem for all eigenvectors takes $O(n^3)$ operations, where n is the number of nodes in the topology. Computing the first k eigenvectors of the Laplacian matrix takes $O(n^3)$ operations and the k-means algorithm takes $O(n)$ operations. Thus, the total cost of our algorithm is $O(n^3)$. This becomes impractical for SDN domain partition where n is the number of network nodes and K is obtained by repeated experiments. Fortunately, our K self-adaptive algorithm can be computed only once. By contrast, the common spectral clustering algorithm needs to repeat experiments to obtain the optimized K, and each experiment takes $O(n^3)$ operations. Obviously, our algorithm is more efficient than the other algorithm.

Aiming to understand the benefits of spectral clustering for the SDN controller placement problem, we have evaluated our algorithms using the Internet2 OS3E topology (https://www.internet2.edu/network/ose/). The OS3E has 34 nodes and 41 edges (Fig. 3).

We implemented a Matlab-based framework to compute the spectral clustering placement results. Fig. 3 shows an SDN domain partition plan based on the spectral clustering algorithm. We can see

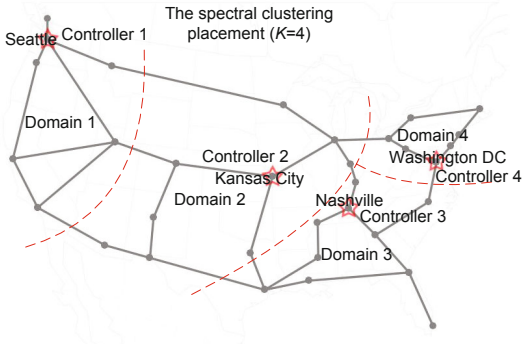

Fig. 3 Partition of four SDN domains based on the spectral clustering algorithm

that the OS3E topology is partitioned into four SDN domains equally when $K = 4$. Among these SDN domains, the controllers will be placed in the nodes that are labeled as stars. As expected, our spectral clustering algorithm meets the requirements of the metrics mentioned in Section 3. From Fig. 3, we can see that the four SDN domains have almost the same size, and that the controller location is close to each clustering center, which meet the balanced partition and average propagation latency metrics.

In the following section, we compare the performance of our placement with other placements mentioned in Heller *et al.* (2012), and design a set of advanced testing scenarios to verify it.

5 Experiments

In this section, we introduce our testing methodology and describe the experimental results of our placement compared with others. All the algorithms mentioned in Section 4 were evaluated with the Beacon controller (https://openflow.stanford.edu/display/Beacon/Home/) and cbench (http://www.openflowhub.org/display/floodlightcontroller/Cbench+(New)). All experiments have been performed on a cluster that consists of 36 machines running 64-bit Ubuntu Server. Each node has two AMD Opteron 2212 2.00 GHz CPUs, 80 GB SCSI HDD, 8 GB RAM, Intel 100 Mb/s Ethernet Controller. In the meantime, we deployed a host as the Beacon controller and others as cbench hosts. The test framework is shown in Fig. 4.

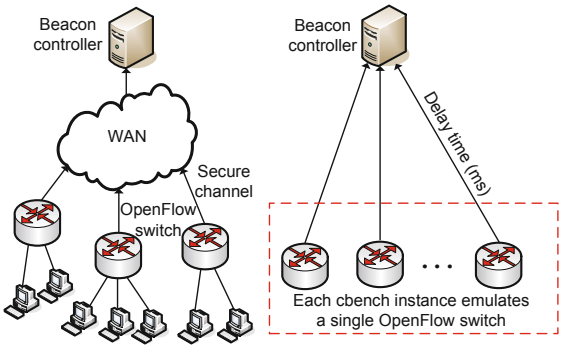

Fig. 4 The cbench emulation for the WAN topology

To evaluate the controllers, we ran Beacon controller software as the WAN SDN controller with the recommended settings, which is a multi-thread

Java-based controller. We relied on the latest available sources of Beacon version 1.04 (April 2014). We chose Beacon because it has better performance than the other controllers (Shah *et al.*, 2013; Shalimov *et al.*, 2013).

We ran cbench instances on multiple modes of the cluster to emulate the switches. As shown in Fig. 4, each cbench instance in our experiments emulated a single OpenFlow switch, and all of these instances sent OpenFlow packet-in messages to a single controller. The cbench instances were connected to the controller with 100 Mb/s interconnects.

To emulate a real network for WAN propagation latencies, we used most nodes of the cluster for running cbench instances. The number of cbench nodes was varied for different experiments, which depended on the metric being calculated. Each cbench emulated a single OpenFlow switch sending packet-in messages to the controller at uniform rates with different delay times. The delay times were calculated by the propagation latencies between two nodes in the WAN topology mentioned above.

To compare the performance of our placement with that of other placements, we also evaluated the average-latency-optimized placement and the worst-case-latency-optimized placement mentioned in Heller *et al.* (2012). The results are shown in Fig. 5.

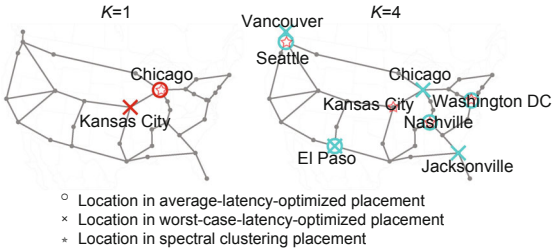

Fig. 5 Three placements for one or four controllers in the OS3E deployment

Fig. 5 shows three placements for $K = 1$ and $K = 4$. The higher density of nodes in the northeast of the US relative to the west leads to a different optimal set of locations for different metrics. The spectral clustering placement is most similar to the average-latency placement and completely different from the worst-case-latency placement. For example, all the controllers of spectral clustering and average-latency placement should go in Chicago when $K = 1$, which balances the high density of the east coast cities with the low density of cities in the west. The different ways produce the same result. However, to minimize the worst-case latency for $K = 1$, the controller should go in Kansas City instead, which is near the center of the topology. As expected, the spectral clustering placement is most similar to the average-latency placement when $K = 4$. By using the mini-max clustering principle, spectral clustering placement can combine latency with performance. The worst-case-latency placement is defined as the maximum node-to-controller propagation delay and is proved to be the least effective method among the three. Thus, we will consider only spectral clustering and average-latency placement in subsequent sections.

Although the placement algorithm (Xiao *et al.*, 2014) may obtain the SDN domain results, it needs to set the number of SDN domains K manually. We suggested the approach mentioned in Section 4 to discover the number of SDN domains by analyzing the eigenvectors. The approach leads to a self-adaptive spectral clustering placement. To evaluate the performance of the approach, we applied it to the OS3E topology. Fig. 6 shows the optimal number of SDN domains K, which is indicated by the point corresponding to the highest eigengap. From Fig. 6, we can see that the corresponding eigengap is maximized when $K = 4$. The results are in agreement with the experimental results obtained by setting K manually, and some results of setting K manually are shown in Table 1. From Table 1, it can be seen that each controller has the best balanced nodes when $K = 4$. Thus, this approach can determine the

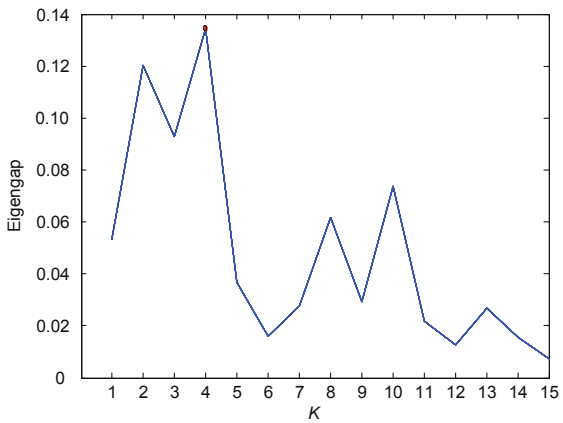

Fig. 6 Eigengap with different *K*'s

optimal number of clusters automatically for spectral clustering placement.

Table 1 The numbers of nodes of SDN domains with different K's

K	C_1	C_2	C_3	C_4	C_5	C_6	C_7
3	14	12	8				
4	7	9	10	8			
5	7	8	3	8	8		
6	7	7	4	7	1	8	
7	7	5	2	2	3	7	8

C_1–C_7: controllers 1–7

To test the effectiveness of our solution, we presented a comparative performance analysis of spectral clustering and average-latency placements. We designed a set of advanced testing scenarios and conducted experiments under many different settings and metrics, which allow us to get a deeper insight into the WAN controller performance issues. All experiments were performed with Beacon and cbench. We ran each experiment five times and took the average number as the result.

5.1 Latency

An important ability of the OpenFlow controller is that it can process the incoming packet-in messages as fast as possible, which we call latency. To measure the controller latencies of the three placements, cbench instances were run in latency mode, in which they generated a packet-in message and waited for a response from the controller before the next packet-in message was sent, and then it counted the total number of responses per second. We kept a cbench instance emulating a single switch, and made many cbench instances send packet-in messages to their controllers with different numbers of connected hosts. Depending on the metric being calculated, the number of cbench instances was varied for different experiments.

For the latency experiments, each test consisted of 500 loops with each lasting 100 ms. The first loop and the last loop were considered as controller warm-up and cool-down, respectively, whose results were discarded. Each test used 100 to 100 000 unique media access control (MAC) addresses (representing emulated end hosts). We kept one worker thread and progressively increased the host density.

Fig. 7 shows the controller latencies of different placements with different numbers of hosts and one thread. In each placement, the controllers were labeled from left to right. For example, spectral clustering placement has four controllers, as Fig. 3 shows. The controller deployed in Seattle was registered under the number '1', the controller deployed in Kansas City under the number '2', and so on. From Fig. 7, it can be seen that most controllers for spectral clustering placement have more balanced responses than average-latency placement. In average-latency placement, the third controller (C'_3) shows the best performance because it serves 13 nodes. However, the second controller (C'_2) serves only three nodes and has exactly the opposite effect. Controller latency is also affected by the propagation latency between the controller and the switch. The first controller deployed in Seattle has lower performance, as Fig. 7 shows, because of the vast distances between the northwest cities.

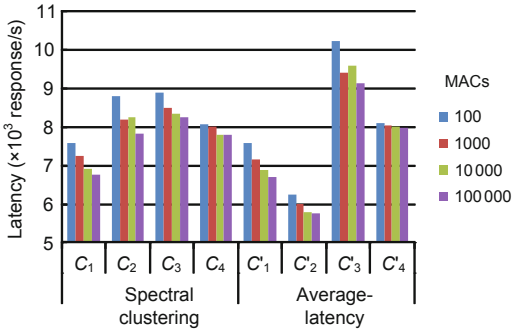

Fig. 7 Latency comparison

To study the impact of propagation latency, we measured spectral clustering placement without delay time. In Fig. 8, we observe that removing the delay time from cbench instances improves the number of responses per second. Moreover, this apparent trend is clear when the propagation latency is larger. The SDN domain 2 for spectral clustering placement has the largest propagation latency. Thus, controller 2 has greatly reduced its response time without delay time. According to the impact of propagation latency, spectral clustering placement is better than average-latency placement. We also find that the other latencies are far less than the propagation latency in the WAN, such as processing latency. Usually, the processing latency has only millisecond-level response time, but the propagation

latency has the second-level response time in WAN. Thus, the propagation latency has the most significant impact on WAN delay.

To test the performance of the two placements under realistic traffic, we conducted the average-latency experiments with the traffic obtained from the wide traces (http://mawi.wide.ad.jp/mawi). The average latency reflects the average response time from each switch to the controller under realistic traffic. The wide trace was obtained from the daily trace at a trans-Pacific line, and demonstrated the features of the links in WAN. From Fig. 9, we can see that spectral clustering placement performs significantly well in terms of average latency under realistic traffic. In average-latency placement, the second controller was deployed in El Paso but not in Kansas City, which leads to the imbalance between the second and third controllers. As expected, the imbalance between the second and third controllers leads to a sharp decline under the realistic traffic. It can be seen that spectral clustering placement shows better performance than average-latency placement under the realistic traffic because of its balanced partitioning.

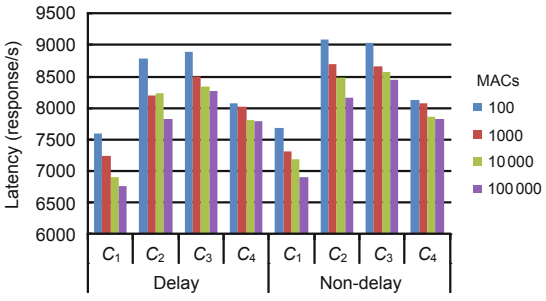

Fig. 8 Latencies with and without delay time

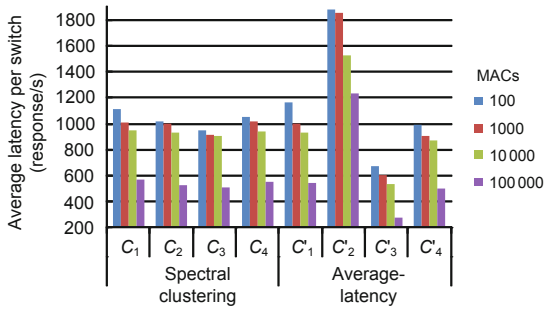

Fig. 9 Average latency comparison with traffic

5.2 Throughput

One of the main objectives for a good controller placement is to minimize the latencies between nodes and controllers in SDN. However, considering only latencies is not sufficient. A placement should also satisfy performance and reliability constraints. In this experiment, we evaluated the effect of controllers in the two placements on the throughput performance, which is the ability to handle a large amount of control traffic. All cbench instances were kept in throughput mode, under which cbench continuously sends packet-in messages to Beacon over a period of time. Our focus in this subsection is to study the average throughput of each controller with different numbers of connected hosts.

For the throughput experiments, each test consisted of five loops with each lasting 1 000 000 ms. The results from the first and the last loops were discarded. The numbers of hosts ranged from 1000 to 1 000 000 in each test. We kept a constant number of eight worker threads and progressively increased the host density. Fig. 10 shows the correlation with the number of connected hosts.

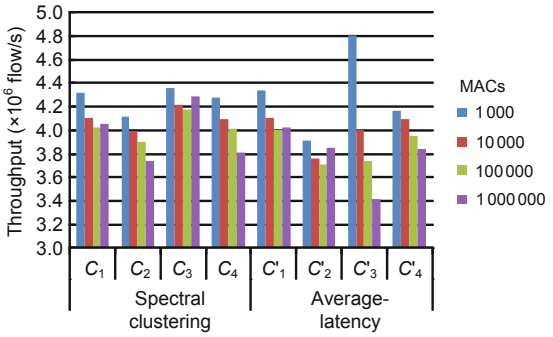

Fig. 10 Throughput achieved with different numbers of hosts with eight threads

The number of hosts in the SDN domain has immaterial influence on the performance of most of the controllers under test. Controller 1 for spectral clustering placement decreased its throughput from 4.3 million to 4.0 million flows per second with 106 hosts. However, in average-latency placement, the performance of controller 3 went down significantly when more hosts were connected. In addition, its controller 2 had the lowest throughput among all controllers. This is caused by the specific details of average-latency placement, namely, the imbalance of

its SDN domain partitioning.

From Fig. 10, it can be seen that spectral clustering placement shows better performance than average-latency placement because of its SDN domain partitioning. We also see an unstable trend in throughput with an increasing number of hosts for average-latency placement.

The performance of an SDN controller is defined by two characteristics: latency and throughput (Shalimov *et al.*, 2013). The goal of SDN controller placement is to obtain the minimum latency and the maximum throughput for each controller. Based on this, we find that spectral clustering placement is more effective than the others.

For the placement, the average throughput reflects the performance and reliability of each controller, which demonstrates significant correlation with the whole network's performance. To find the impact of K on the network's performance, we tested the average throughput under the wide traces (http://mawi.wide.ad.jp/mawi) with different values of K. As shown in Fig. 11, the placement performed significantly well and the average throughput changed more gently with a growing number of hosts when $K = 4$, which agrees with the conclusions drawn from Table 1. From Fig. 11, we can also see that the average throughputs of other placements dropped rapidly because of their imbalanced nodes.

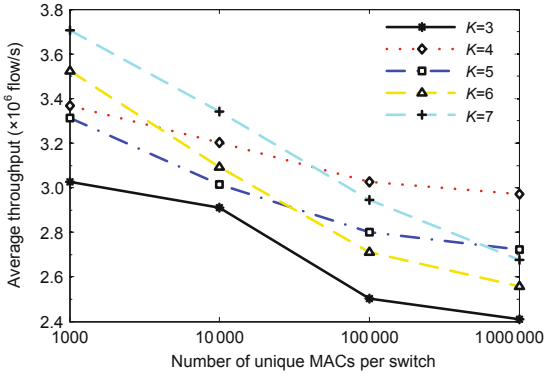

Fig. 11 Average throughput comparison with different K's

5.3 Reliability

Reliability is the ability of the controller to work normally over a long period under an average workload. To evaluate the reliability, we measured the number of failures during a long time period under a given heavy workload. In this experiment, we kept a constant number of eight worker threads for each controller, but increased the total number of packet-in messages from the cbench instances running on each node. In our test case, we used 1 000 000 unique MAC addresses per switch for the stress tests, and each switch sent OpenFlow packet-in messages at rates varying from 1000 to 10 000 requests per second. All tests were run for 24 h and the number of errors was recorded during the test. By error, we mean either a failure to receive a reply from the controller or an input/output (I/O) error from the Beacon buffer.

The experiments have shown that most of the controllers successfully coped with the test load, except the third controller for average-latency placement. The third controller for average-latency placement dropped 53 241 567 messages and closed 179 connections. For average-latency placement, the third controller's failures were caused by serving too many nodes, which leads to the instability of average-latency placement. We also found that the controller was unstable when it served more than 11 nodes in our tests. Compared with the deployment in LAN, the reliability of the controller deployed in WAN declined greatly.

To verify the applicability and effectiveness of spectral clustering placement, we expanded our analysis to more topologies in the Internet Topology Zoo (Knight *et al.*, 2011). The Internet Topology Zoo covers a diverse range of geographic areas, network sizes, and topologies. The graphs in the Zoo do not conform to any single model, and can be used to verify the applicability of our approach. In most cases, we can easily obtain the balanced cut by using spectral clustering placement. We also find that the correct number of clusters is important for spectral clustering placement. When the network has more than 100 nodes, prior knowledge of the number of clusters is required.

6 Conclusions

In this paper, we have proposed a K self-adaptive SDN controller placement for WAN. Our approach is based on partitioning a large network into several small SDN domains by using the

spectral clustering placement algorithm. To maximize the reliability of the controller and to minimize the latency of WAN, we have presented the metrics for spectral clustering placement. We have suggested exploiting the structure of the eigenvectors to determine automatically the number of SDN domains. As a result, a self-adaptive spectral clustering algorithm based on the matrix perturbation theory has been proposed. After presenting a test framework with Beacon and cbench, the ideas and mechanisms were illustrated by using the Internet2 OS3E topology. We conducted experiments under many different settings and metrics. Experimental results showed that self-adaptive placement is good at solving the SDN controller placement problem and determining the number of SDN domains automatically.

However, we noted that understanding the overall SDN controller placement remains an open research problem. The placement is likely a complex function of the topology, metric, and the value of K. Our approach presented in this paper is just a first step towards the SDN domain partition. In future work, we expect to expand our analysis to other network latencies.

References

Bach, F.R., Jordan, M.I., 2003. Learning Spectral Clustering. Technical Report, No. UCB/CSD-03-1249. University of California at Berkeley, USA.

Cai, Z., Cox, A.L., Ng, T.S.E., 2010. Maestro: a system for scalable OpenFlow control. Technical Report, TR10-08. Rice University, USA.

Dixit, A., Hao, F., Mukherjee, S., *et al.*, 2013. Towards an elastic distributed SDN controller. *ACM SIGCOMM Comput. Commun. Rev.*, **43**(4):7-12. http://dx.doi.org/10.1145/2491185.2491193

Erickson, D., 2013. The beacon OpenFlow controller. Proc. 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.13-18. http://dx.doi.org/10.1145/2491185.2491189

Gude, N., Koponen, T., Pettit, J., *et al.*, 2008. NOX: towards an operating system for networks. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(3):105-110. http://dx.doi.org/10.1145/1384609.1384625

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. Proc. 1st Workshop on Hot Topics in Software Defined Networks, p.7-12. http://dx.doi.org/10.1145/2342441.2342444

Hock, D., Hartmann, M., Gebert, S., *et al.*, 2013. Pareto-optimal resilient controller placement in SDN-based core networks. Proc. 25th Int. Teletraffic Congress, p.1-9. http://dx.doi.org/10.1109/ITC.2013.6662939

Kirkpatrick, K., 2013. Software-defined networking. *Commun. ACM*, **56**(9):16-19. http://dx.doi.org/10.1145/2500468.2500473

Knight, S., Nguyen, H.X., Falkner, N., *et al.*, 2011. The Internet topology zoo. *IEEE J. Sel. Areas Commun.*, **29**(9):1765-1775. http://dx.doi.org/10.1109/JSAC.2011.111002

Koponen, T., Casado, M., Gude, N., *et al.*, 2010. Onix: a distributed control platform for large-scale production networks. Proc. OSDI, p.1-14.

Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., *et al.*, 2015. Software-defined networking: a comprehensive survey. *Proc. IEEE*, **103**(1):14-76. http://dx.doi.org/10.1109/JPROC.2014.2371999

Lin, P., Bi, J., Wang, Y., 2013. East-west bridge for SDN network peering. Proc. 2nd CCF Int. Conf. of China, p.170-181. http://dx.doi.org/10.1007/978-3-642-53959-6_16

Liu, N., Lu, Y., Tang, X.J., *et al.*, 2014. Study on automatically determining the optimal number of clusters present in spectral co-clustering documents and words. *J. Chin. Comput. Syst.*, **35**(3):610-614 (in Chinese).

Mall, R., Langone, R., Suykens, J.A.K., 2013. Self-tuned kernel spectral clustering for large scale networks. Proc. IEEE Int. Conf. on Big Data, p.385-393. http://dx.doi.org/10.1109/BigData.2013.6691599

McKeown, N., Anderson, T., Balakrishnan, H., *et al.*, 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(2):69-74. http://dx.doi.org/10.1145/1355734.1355746

Ng, A.Y., Jordan, M.I., Weiss, Y., 2001. On spectral clustering: analysis and an algorithm. *In*: Dietterich, T.G., Becker, S., Ghahramani, Z. (Eds.). Advances in Neural Information Processing Systems 14, p.849-856.

Phemius, K., Bouet, M., Leguay, J., 2014. DISCO: distributed multi-domain SDN controllers. Proc. IEEE Network Operations and Management Symp., p.1-4. http://dx.doi.org/10.1109/NOMS.2014.6838330

Rebagliati, N., Verri, A., 2011. Spectral clustering with more than K eigenvectors. *Neurocomputing*, **74**(9):1391-1401. http://dx.doi.org/10.1016/j.neucom.2010.12.008

Shah, S.A., Faiz, J., Farooq, M., *et al.*, 2013. An architectural evaluation of SDN controllers. Proc. IEEE Int. Conf. on Communications, p.3504-3508. http://dx.doi.org/10.1109/ICC.2013.6655093

Shalimov, A., Zuikov, D., Zimarina, D., *et al.*, 2013. Advanced study of SDN/OpenFlow controllers. Proc. 9th Central & Eastern European Software Engineering Conf. in Russia, Article 1. http://dx.doi.org/10.1145/2556610.2556621

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell.*, **22**(8):888-905. http://dx.doi.org/10.1109/34.868688

Tam, A.S.W., Xi, K., Chao, H.J., 2011. Use of devolved controllers in data center networks. Proc. IEEE Conf. on Computer Communications Workshops, p.596-601. http://dx.doi.org/10.1109/INFCOMW.2011.5928883

Tian, Z., Li, X., Ju, Y., 2007. Spectral clustering based on matrix perturbation theory. *Sci. China Ser. F*, **50**(1):63-81. http://dx.doi.org/10.1007/s11432-007-0007-8

Tootoonchian, A., Ganjali, Y., 2010. HyperFlow: a distributed control plane for OpenFlow. Proc. Int. Network Management Conf. on Research on Enterprise Networking, p.1-6.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., *et al.*, 2012. On controller performance in software-defined networks. Proc. 2nd USENIX Conf. on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, p.1-6.

von Luxburg, U., 2007. A tutorial on spectral clustering. *Stat. Comput.*, **17**(4):395-416. http://dx.doi.org/10.1007/s11222-007-9033-z

Wang, L., Bo, L.F., Jiao, L.C., 2007. Density-sensitive spectral clustering. *Acta Electron. Sin.*, **35**(8):1577-1581 (in Chinese).

Wauthier, F.L., Jojic, N., Jordan, M.I., 2012. Active spectral clustering via iterative uncertainty reduction. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1339-1347. http://dx.doi.org/10.1145/2339530.2339737

Xiao, P., Qu, W., Li, Z., 2014. The SDN controller placement problem for WAN. Proc. IEEE/CIC Int. Conf. on Communications in China, p.220-224. http://dx.doi.org/10.1109/ICCChina.2014.7008275

Xie, H., Tsou, T., Lopez, D., *et al.*, 2012. Software-Defined Networking Efforts Debuted at IETF 84. Available from http://www.internetsociety.org/articles/software-defined-networking-efforts-debuted-ietf-84.

Yin, H., Xie, H., Tsou, T., *et al.*, 2012. SDNi: a Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains. Available from https://tools.ietf.org/html/draft-yin-sdn-sdni-00.

Yu, M., Rexford, J., Freedman, M.J., *et al.*, 2010. Scalable flow-based networking with DIFANE. *ACM SIGCOMM Comput. Commun. Rev.*, **40**(4):351-362. http://dx.doi.org/10.1145/1851275.1851224

Zelnik-Manor, L., Perona, P., 2004. Self-tuning spectral clustering. *In*: Saul, L.K., Weiss, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 17, p.1601-1608.