A Multi-Domain SDN Scalability Architecture implementation based on the Coordinate Controller

Jianglong Wang, Guochu Shou, Yihong Hu, Zhigang Guo
Beijing Laboratory of Advanced Information Networks
School of Information and Communication Engineering
Beijing University of Posts and Telecommunications
Beijing 100876, China
{jianglong93, gcshou, yhhu, gzgang}@bupt.edu.cn

***Abstract*—Software-defined Networking, known as SDN, is a novel architecture that decouples the control plane from the data plane, and opens a programmable network. However, it is common practice for network operators to divide the large-scale networks into multiple domains and manage the networks by using a group of controllers. Designing a scalable SDN control plane becomes a critical problem. In this paper, we propose a SDN scalability architecture for multi-domain, multi-vendor networking. We design and implement the Coordinator Controller to enable different SDN administrative domains to cooperate by unifying the northbound APIs of the controllers which are working for different SDN domains. To validate the proposed architecture, we build a multi-domain experiment environment consisting of three vendors. The results show great ability in the network state view consistency and end-to-end provisioning services.**

Keywords-SDN scalability; Coordinate Controller; multi-vendor; multi-domain;

I. INTRODUCTION

With the development of the network services and web applications, current network architecture which is characterized by closed system does not favor network innovation. Convergence, openness are the key features of the future network [1]. SDN provides a programmable network over abstracted network resources, by means of separating the control plane from data plane and utilizing the control plane in a centralized controller [2]. SDN is a well-acknowledged technology supporting dynamic features of future network and intelligent applications.

In the openflow-based SDN network, controller is used to compute packet flow route path, generate flow tables, and put them into the corresponding switches through openflow protocol [3]. However, in the certain scale network, every 100 switches in the data center can produce 10000K request per second [4], but NOX controller can only handle 30K flow per second [5]. As the newly launched service flow initial request is always sent to the controller, there must be a serious problem about the response time of the new packet flow [6]. Meanwhile, a single controller would be likely to put the whole network at risk due to the single point of failure [7]. Thus, as the number of SDN deployment scenarios extends to the large scale networks, which consist of multiple domains with different administrative vendors or any other heterogeneous network, using multiple controllers to manage the network becomes a common approach. The concerns about SDN scalability are gradually introduced to a multiple controller collaboration statement [8].

In this paper, we will have a comprehensive discussion on the SDN scalability architecture and its latest progress. Then, we will introduce a novel vendor-agnostic Coordinate Controller to make SDN controllers work together. In order to allow the Coordinate Controller to interact with SDN controllers, we design the information should be exchanged among domains and unify SDN domain controllers' northbound APIs to share this information. More detailed implementation plans are given, and we test this architecture in multi-vendor SDN networking environment, and do the northbound APIs (NBI) interoperability test, network state view consistency test, end-to-end provisioning services test over this vendor-agnostic Coordinate Controller.

II. BACKGROUND AND RELATED WORK

A. Scalability Architecture

As it mentioned before, the SDN starts with a single controller that would be responsible for the whole network, such as NOX, POX. Although multi-core controller demonstrates superior performance than traditional single controller, it is still confronted with many barriers in its function and performance limitation. A multi-controller solution for control plane becomes necessary indeed [8]. Generally, SDN scalability architecture can be classified into two types: horizontal and hierarchical structure.

Horizontal structure is that all of controllers should be in the same level and communicate with each other in peer-to-peer way through west-east interface. They have to share the whole network information with each other to construct a consistent state view. As the network topology changes, all controllers will be synchronized its state with each other. Thus, achieving consistence for each controller is under great consideration, west-east interface plays an important role in controller peers' communication. To address this problem, researchers propose West-East Bridge [9] for SDN networks peering. The west-east interface can exchange the inter-domain information among controllers and the West-East Bridge should be compatible with different third-part controllers and network view storage systems.

978-1-5090-5154-0/16 $31.00 © 2016 IEEE
DOI 10.1109/CyberC.2016.100

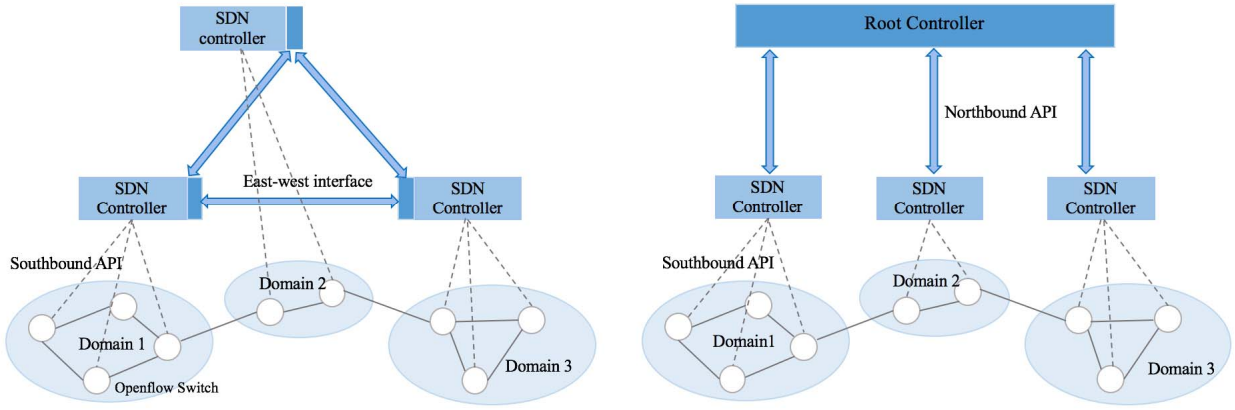

Fig. 1 Horizontal structure (left) and hierarchical structure (right) of SDN scalability architectue

Onix [10] is the first SDN distributed controller, it has a network information base (NIB) to manage its controllers. As long as it keeps the consistency of NBI, the controllers can be operated under a global view of the network. It also provides generic distributed state management APIs, programmers can program their control logic conveniently. HyperFlow [11] is comprised with many distributed controllers, which communicate with each other by registering and broadcast mechanism. It is an APP implementation for NOX controller. Moreover, when a controller instance fails, HyperFlow can get the switches which are under failed controller reconfigure to the new controller, exhibiting a good availability in failover purposes.

ONOS and Opendayligtht are the most popular controller in the open source community. ONOS [12] provides a distributed core to maintain a global view for every controller instance running on different servers. The Multi-Clusters Peering Provider in ONOS is a west-east interface used to help deploying ONOS clusters cross-domain services. Opendaylight (ODL) [13] is also able to build consistent clustering, and provides fine-grained redundancy and scalability on the condition of state consistency. Among all of the released versions, ODL Li controller adds the application of SDNi protocol [14], which enables ODL to exhibit high scalability in heterogeneous multi-controller cooperation. SDNi is a west-east protocol used to communicate between multiple SDN domains, and it defines the Reachability, flow setup/tear down/update, Capability and other arguments as the interaction information. This kind of interoperability can extend the control plane effectively.

Hierarchical structure is another scalability architecture and it organizes controllers in a tree structure. Controllers are divided into two different types, one root controller and multiple leaf controllers. The root controller which is placed in the top layer, acting as a network operating system in the entire network, has the ability to allocate the whole network resources. A leaf controller only needs to manage its local domain, and maintains the local network view directly. Compared with the horizontal way, the most distinguished feature of hierarchical way is that the local controller is no longer achieve the complete network information. In this way, when a service flow sends a request, it can ensure that at least one controller can response to it immediately. So, it is a better way to improve the performance of multi-controller networking, especially in a large scale network.

Kandoo [15] is a representative of hierarchical structure, which has two layers of controllers in control plane, root controller and local controller, and the information interaction between them is based on the APP reroute agent running in the Root controller and APP detect agent running in the Local controller. When the switch is forwarding packets, the APP detect application of the nearest controller will detect the elephants stream needed to be reported to the root controller, and the APP reroute application in the root controller will complete the deployment of network-wide services. This approach avoids the frequent interaction with root controller, effectively reducing the traffic load, greatly enhancing performance efficiency.

B. Issues in Multi-domain SDN

In general case, it is common practice to divide the network into multiple domains, and each domain provides different control plane technologies with specific controller by different vendors [16]. Clearly both horizontal and hierarchical scalability architecture can be used to implement multi-domain SDN networking, but there is a potential problem about how to maintain consistency of the global network view in a distributed control system [17]. Providing a strictly consistent view will lead to high network overhead due to the frequency of synchronized messages among controllers, whereas achieving a higher responsiveness should be at the expense of weakly inconsistent network state view. Meanwhile, different domains belong to different administrative carriers, they must have some secret information not be shared with each other, so it is hard to implement the equally communication among multiple domains in the same horizontal level.

In this paper, we leverage the hierarchical architecture and propose a Coordinate Controller in the face of heterogeneous networked system consisting of multi-vendors and multi-domains. The key features for the Coordinate Controller are in the following ways:

- Coordinate Controller can present a global consistent network topology view.
- Allow end-to-end provisioning services across multiple domains.
- Unify the vendor-agnostic northbound APIs of local domain controllers to uniformly interact with Coordinate Controller.

III. Coordinate Controller

Fig.2 shows the Multi-domain SDN scalability architecture based on the Coordinate Controller. Similar to traditional SDN, the proposed architecture is divided into 3 planes: Application Plane, Control Plane and Data Plane. The data plane consists of SDN-enabled switches, subdivided into administrative domains. The control plane is divided into 2 sub-planes: Coordinate Controller layer and Local Domain Controller layer. Coordinate Controller is acting as the unified network operating system in the whole network, allowing a complete and higher-level network resources abstract, and providing end-to-end services across multi-domain regardless of the different control plane technologies deployed in each domain. It is responsible for the global network information collection, resource allocation, domain information interaction and so on. By contrast, Local Domain Controller is a traditional SDN controller which is only used to manage network for each region, it can be provided by different administrative vendors, open-source controllers such as Opendaylight, ONOS, Ryu, are also available. The interface between Coordinate Controller and Local Domain Controller are the restful APIs we designed to abstract and orchestrate the control plane functions of SDN controllers. The APIs are open but in the form of standard way. Thus, all of the domain SDN controllers should implement the unified NBI. Certainly, the APIs are extensible, we can implement them according to the service orchestration need. The more detailed information about this architecture are as followed.

A. Architecture

The Coordinate Controller architecture model, conveyed in Fig.1, is mainly composed of 5 modules:

- Protocol Interpreter: control the communication between Coordinate Controller and Local Domain by dealing with the unified restful API or openflow protocol.
- Topology Manager: manage the network topology and resources reported by Local Domain Controller, including inter-domain topology and resource.
- Route Manager: select the proper route for requested service, including the inter-domain path calculation and cross-domain path calculation.
- Connection Manager: build, modify and delete a service connection.
- Resource Virtualization Manager: abstract the topology and network resource information from the local domain controllers and provide the abstract virtualized information to SDN Applications over this Coordinate Controller.

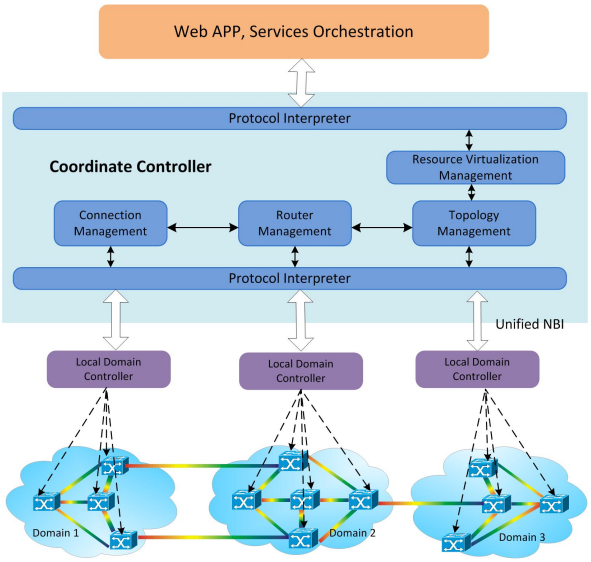

Fig.2 Multi-domain SDN scalability architecture and Coordinate Controller

B. End-to-end provisioning services

Coordinate Controller has the ability to implement end-to-end provisioning services orchestration across multiple domains. Fig.3 shows the process of deploying a new service. When a SDN application request resources for a new service, these elements will work together to provide resources for the application. Firstly, applications should send the request to ask controller for a new service. Then the Protocol Interpreter will translate this request into source and destination address to Connection Manager. Next, Connection Manager sends these two addresses to Route Manager for calculating. At last Protocol Interpreter will send the result to application and post path information of each domain to Local Domain Controller. In this way, Coordinate Controller can manage the process of creating a new service and help the distributed domains work together.

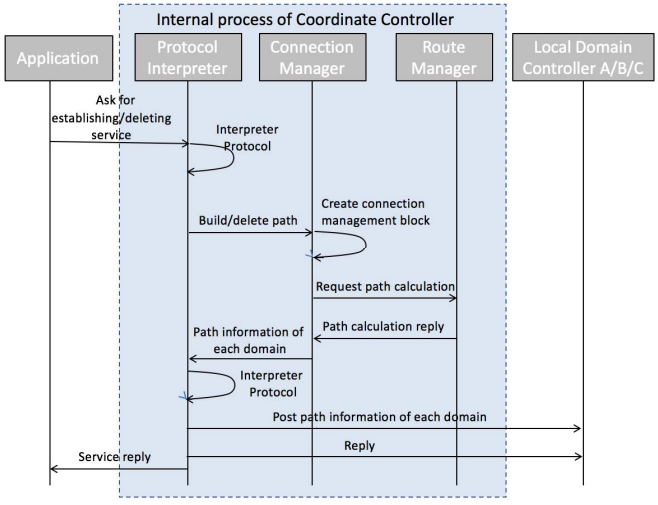

Fig. 3 The process of delopying a new end-to-end provisioning services

C. *Unified vendor-agnostic NBI design*

To solve the problem about the diversity of vendors, we design an open, vendor-agnostic northbound API (NBI) for Local Domain Controllers. The APIs abstracts a set of control plane function and determine what domain information needed to be share. They are in line with the latest OIF/ONF Transport API which concentrate on the standardization efforts for orchestration of REST API for SDN controllers [18]. With this unified NBI, Local Domain Controller can provide the abstracted resource for the Coordinate Controller. Fig. 4 shows the local domain SDN controller structure. Our experimental unified vendor-agnostic NBI are divided into two parts, Topology API and Service API.

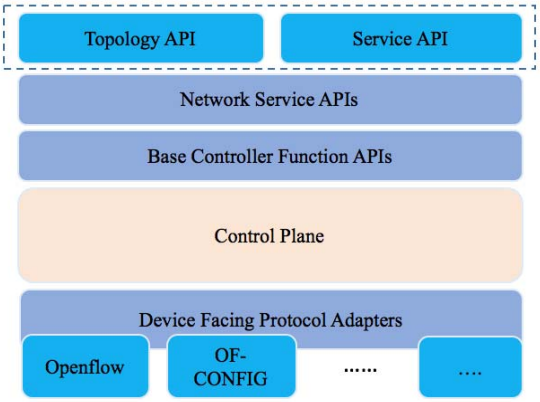

Fig. 4 Local Domain Controller Stucture

1) *Topology API:*

Topology API is responsible for gathering network elements from each domain to construct a global view of the whole network. A Topology object involves a set of nodes (Vertex, API description) and their connection links (Edge), A Vertex object consists of a set of available ports (Edge End) and the available network resources, such as bandwidth (Edge end resource). At the same time, Topology API also defines many actions for operating topology object, Creation, List, Read, Retrieval, Deletion and so on.

2) *Service API:*

Service API aims to launch a service request and set up an end-to-end connection within the network. In the Service API, Call Object is defined as the request interface, which stands for a new service request, including the link connection between source node and destination node (referred to aEnd and zEnd in API description). Then, the path computation unit will give a path calculation between two nodes and return a Path object containing information about the route path between two nodes.A Connection object is responsible for the configuration of the datapath by a local domain controller entity. With the Service API, we can easily set up an end-to-end network and connectivity services across multiple domains.

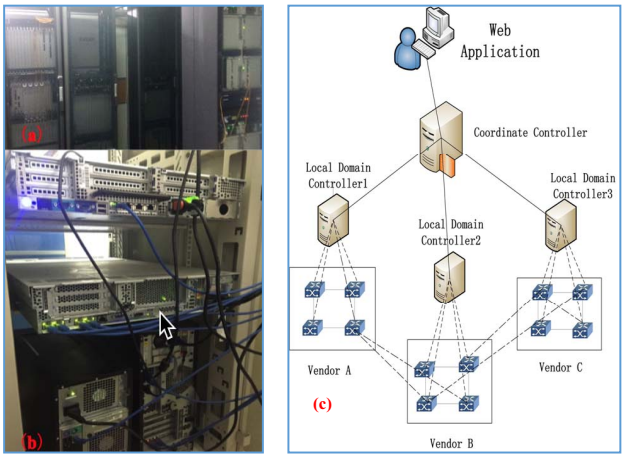

Fig. 5 (a) Network equipments of the vendors. (b) Servers for Local Domain Controller and Cooridnate Controller. (c) Multi-vendor networking topology diagrammatic.

IV. Multi-vendor SDN networking Experiment

ONF and OIF have launched the Global Transport SDN Prototype to promote the development of transport SDN [17]. There were nine vendors and five carrier labs participating in the prototype implementation and interoperability test. This paper describes the work done in one of these labs.

To experimentally evaluate the proposed architecture, we build a heterogeneous multi-domain experiment environment, which is comprised of three vendors, we call vendor A, B, C for short. They have different controllers using the unified NBI to control local domain network elements. We implement the Coordinate Controller in a server to manage three Local Domain Controllers. The real scenes are shown in Fig. 5(a) (b). The networking topology is shown in Fig. 5(c).

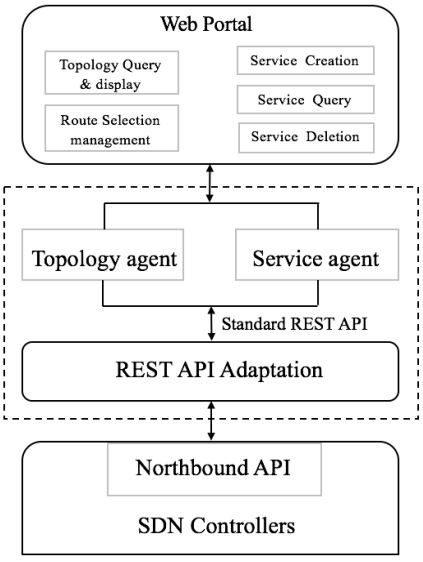

Fig. 6 Web APP Stucture

We also design a testing Web APP to accomplish the test of the architecture, which acts as a network orchestrator. The structure of the Web APP is shown in Fig. 6. The APP is used to present the global topology view abstracted by Coordinate Controller, and provide the portal for topology information query, service creation, query and deletion. Route management module is also embedded in this testing APP, which applies to establish different types of service. The web backstage consists two mainly component, topology agent and service agent are function libraries that used to deal with Topology API events and Service API events.

A. NBI interoperability test

First, we take the NBI interoperability test [19] for local domain controllers which come from different vendors. Because the interfaces of domain controllers are designed in a restful style, we send http requests to domain controllers in the Web APP, then verify whether the Json-format response meets the standards. Test cases for Topology API and Service API are described in table I.

The result shows all the participating vendors have implemented the unified northbound interface in their controllers, and demonstrate great interoperability under the control of the Coordinate Controller.

TABLE I. TEST CASE

Type	Description
Topology API test	Retrieve vertex list;
	Read vertex detailed information;
	Retrieve edge end list;
	Read a specific edge end information;
	Read edge end resource transfer;
	Retrieve edge list;
	Read edge representation;
	Read all resource representations at once.
Service API test	Create a cross-domain service connection between two edge ends;
	Retrieve lists of services and detailed information from all domains.
	Delete a cross-domain service.

B. Network state view consistency test

Coordinate controller acts as the unified network operating system in the top layer of the control plane, and it should have the whole network information. Thus, maintaining network state view consistency is the primary condition for supporting heterogeneous network services. When getting the global topology from local domains through Topology API, the web APP will display the topology view as well as the properties of each network element. The topology view is showed in Fig. 7, and it can prove that Coordinate Controller has the consistent network state view in a global perspective.

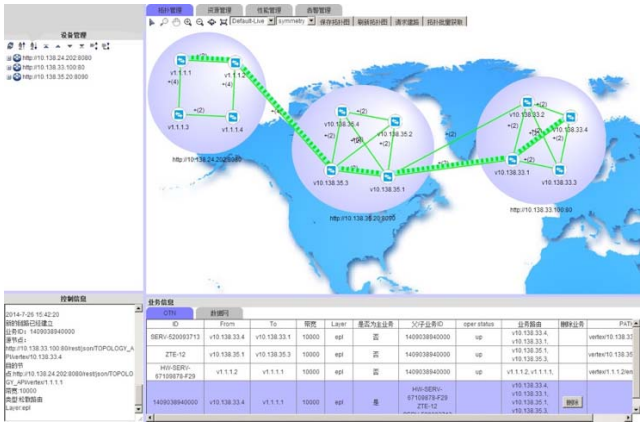

Fig. 7 The global network view in web APP demostration

C. End-to-end provisioning services test

In the Web App, we select any port of the node as the source and destination node in the Service Creation module, and send a service request to create a connection to Coordinate Controller. The coordinate controller conducts the path computation, then notifies a local domain controller to complete service deployment by configuring flows tables into the corresponding switches.

In the process of establishing the end-to-end service, the route selection management module plays an important role in routing-related function, including the calculation of cross-domain path and inter-domain path. In this experiment, we classify the route into two types, loose routing and strict routing, the path computation is based on the shortest path algorithm. Loose routing which is calculated by Coordinate Controller, only gives the output port and input port of each domain, the inter-domain path between the input and output port is calculated completely by the Local Domain Controller. In this way, we can reduce the overload of the Coordinate Controller to improve the efficiency of the services establishment. Strict routing refers directly by the Coordinate Controller which gives every connection node in each domain and passes the path information to the Local Domain Controller. It makes a big difference in the service scene which requires specific nodes or paths. The Coordinate Controller plays a significant role in end-end provisioning services.

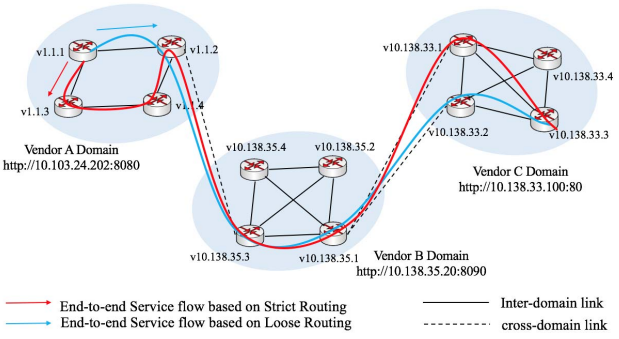

Fig. 8 End-to-end service flow based on strict routing and loose routing.

We use the Web APP to launch a new end-end service request to Coordinate Controller, after path computation, Coordinate Controller will send a service reply including the final path to Web APP which displays the path on the screen as shown in Fig. 7. We also test the end-to-end service establishment based on strict routing and loose routing policy. Fig. 8 is the schematic diagram of the service flow based on the strict routing and loose routing. The result shows that the Coordinate Controller enables the end-to-end provisioning services across multi-domain.

V. CONCUSION

In this paper, we have a comprehensive discussion on the SDN scalability architecture, and propose a Coordinate Controller for multi-domain SDN networking based on the hierarchical structure. We test this architecture in multi-domain SDN networking environment comprised with three vendors. The experiment results demonstrate the Coordinate Controller can keep consistency in network state view, and have the ability to establish end-to-end provisioning services across multi-domain. At the same time, we define the unified vendor-agnostic northbound APIs for each Local Domain Controller, and it demonstrates great interoperability over the controller provided by multi-vendor, which will greatly facilitate the collaboration for multi-vendor services and the standardization progress of SDN northbound interface.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (Grant No. 61471053), and project of Beijing Laboratory of Advanced Information Networks.

REFERENCES

[1] Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

[2] N.Mckeown, “Keynote talk:software-defined networking,” Proc. IEEE INFCOM, April 2009.

[3] N.Mckeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Rexford, S.Shenker, J.Turner, “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, 2008, pp. 69−74.

[4] Z.Cai, et al, “Maestro,a system for scalable Open-Flow control,” Techical Report, TR10-08, Rice University, 2010.

[5] Gude N, Koponen T, Pettit J, et al, “NOX: towards an operating system for networks,” ACM SIGCOMM Computer Communication Review, 2008, pp. 105-110.

[6] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, “On scalability of software-defined networking,” in IEEE Communications Magazine, vol. 51, no. 2, pp. 136-141, February 2013

[7] D.Kreutz, F.Ramos, and P.Verissimo, “Towards secure and dependable software-defined networks,” ACM SIGCOMM Workshop, Mar. 2013.

[8] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, “On scalability of software-defined networking,” in IEEE Communications Magazine, vol. 51, no. 2, pp. 136-141, February 2013.

[9] P. Lin, J. Bi, Z. Chen, Y. Wang, H. Hu and A. Xu, "WE-bridge: West-east bridge for SDN inter-domain network peering," Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on, Toronto, ON, 2014, pp. 111-112.

[10] T. Koponen et al, ‘‘Onix: A distributedcontrol platform for large-scale productionnetworks,’’ in Proc. 9th USENIX Conf. Oper.Syst. Design Implement, 2010, pp. 1–6.

[11] A.Tootoonchian, Y.Ganjali, “HyperFlow: a distributed control plane for OpenFlow,” Proceedings of the 2010 internet network management conference on Research on enterprise networking, 2010, pp. 3-3.

[12] P.Berde, M.Gerola, J.Hart, et al, “ONOS: towards an open, distributed SDN OS,” Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1-6.

[13] OpenDaylight, http://www.opendaylight.org.

[14] H. Yin et al, ‘‘SDNi: A message exchange protocol for software defined networks(SDNS) across multiple domains,’’ InternetEngineering Task Force, Internet Draft, Jun. 2012. http://tools.ietf.org/id/draft-yin-sdn-sdni-00.tx

[15] S.Hassas Yeganeh, Y. Ganjali, ‘‘Kandoo: a framework for efficient and scalable offloading of control applications” Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 19-24.

[16] Zhengxin Guo,Yihong Hu, Guochu Shou, Zhigang Guo, “An Implication of Multi-domain Software Defined Networking,” Wireless Communications, Networking and Mobile Computing, Wicom 2015, 2015:588 - 592.

[17] L.Dan, A.Wundsam, B.Heller, et al, “Logically centralized? State distribution trade-offs in software defined networks,” Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012, pp. 1-6.

[18] OIF/ ONF White Paper, Global Transport SDN Prototype Demonstration, October 7, 2014.

[19] Ruixue Gong, Guochu Shou, Yihong Hu, Zhigang Guo, Guoying Zhang and Hui Ding, “Interoperability test of northbound interface for a transport SDON prototype,” Opto-Electronics and Communications Conference (OECC), 2015, Shanghai, 2015, pp. 1-3.