# EWBridge: East-West Bridge for Heterogeneous SDN NOSes Peering

Pingping Lin, Jun Bi, Yangyang Wang
Network Research Center, Department of Computer Science, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology
4-204, FIT Building, Beijing, 100084, China

linpp@netarchlab.tsinghua.edu.cn, junbi@tsinghua.edu.cn, wyystar@gmail.com

## ABSTRACT

The east-west interface between controllers is private in the current SDN distributed network operating systems. Different controllers cannot work together. This paper designs a high-performance East-West communication system called EWBridge for heterogeneous NOSes to exchange network view and work together.

Our proposal is to deploy the EWBridge system to three SDN networks: SDN networks in CERNET, INTERNET2, and CSTNET. Based on this, we can run a path commuting application on each NOS to let the three SDN networks peering and communicating.

## 1. INTRODUCTION

When deploying SDN [1] in real networks, large networks are always partitioned by the network operators into several sub-networks. Each sub-network runs one NOS or controller such as NOX [4], Floodlight [3], etc. Each NOS just has a local network view. However, many network applications (APPs) need a global view of the entire network rather than just one sub-network information. Thus, multiple NOSes may need to communicate with each other to exchange individual network view information or share a global network view database to construct a global network view.

There are several distributed NOSes such as Onix [6], HyperFlow [7]. However, none of them can coexist with others, because the east-west communication interface is private. Our target is to enable different NOSes from different vendors to co-work together.

However, what network information and how such information should be exchanged has not been well addressed so far, especially when there exist multi-vendor NOSes. We refer to this problem as the East-West Communication Problem. East-West Bridge (EWBridge) is proposed for the multiple heterogeneous NOSes working together.

## 2. EAST-WEST BRIDGE FOR SDN PEERS

We designed a high-performance mechanism named EWBridge shown in Figure 1 for heterogeneous NOSes to exchange network views. It defined what information should be shared and how it should be shared. Enabled by the characteristic of JSON (JavaScript Object Notation) (Other similar languages are eXtensible Markup Language, YANG [5], YAML [2]), EWBridge supports NOSes from different third-party vendors.

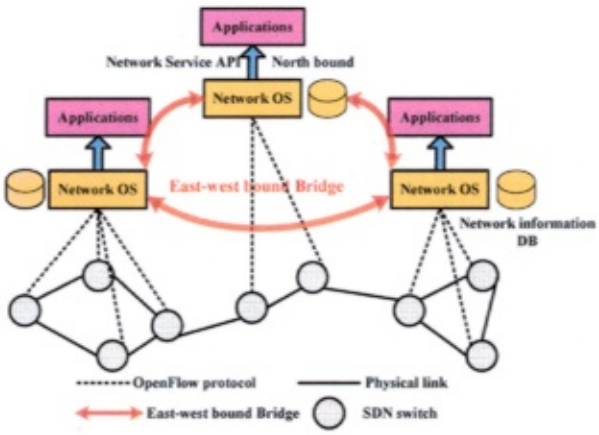



Fig. 1. East-west Bridge for heterogeneous NOSes

### 2.1 Network View Abstraction

A network view mainly includes two aspects: the static network view and the dynamic view. The static network view includes the following information: *a) Reachability:* in carrier network, reachability refers to the IP address prefixes; in DC and

enterprise network, it also includes the server/host addresses. Reachability is the least information that SDN peers should exchange; *b) topology:* node (e.g., switch, server, host, controller, even firewall, balancer, others), link, link bandwidth, port throughput, link connection; *c) network service capabilities:* such as SLA (Service Level Agreement), GRE (Generic Routing Encapsulation), and SSL (Secure Sockets Layer); *d) QoS parameters*, such as latency, reliability, packet loss rate, availability, throughput, time delay variation, and cost. While the dynamic network view mainly includes the network status, such as FlowTable entries information in each switch, real-time bandwidth utilization in the topology, and all the flow paths in the network.

## 2.2 High Speed Bridge Exchange Mechanism

All the controllers can establish a virtual full mesh topology based on TCP (Transmission Control Protocol). All the SDN peers are equal to each other. For the network event such as link failure, adding/deleting switch, adding/deleting IP prefixes, each controller can subscribe to other controllers' database events. EWBridge adopts the publish/subscribe system to deliver update messages. Once an event is triggered, the corresponding controller will push the event to all the subscribers simultaneously. Each controller can get the update message directly from the controller it cares.

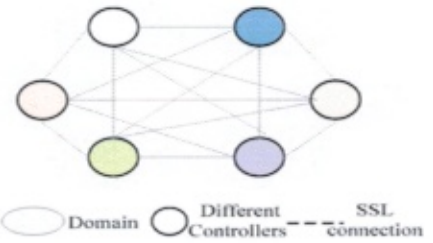
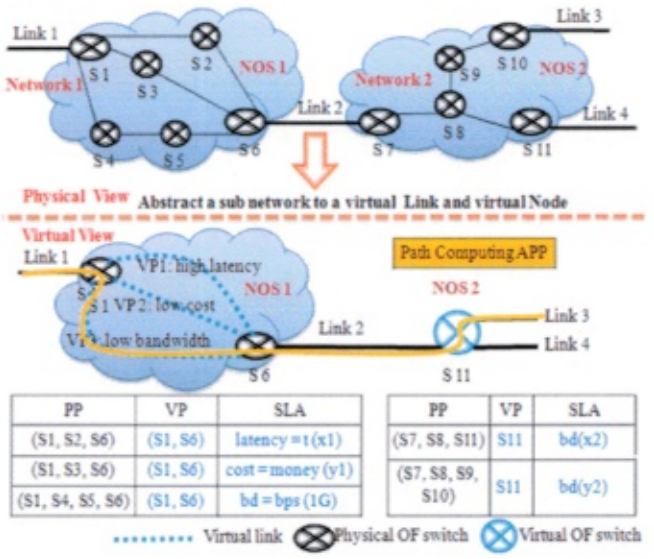



Fig. 2. TCP based full mesh topology for EWBridge

The EWBridge is a stripped-down version of the BGP protocol. Compared with the BGP protocol, EWBridge protocol changed the UPDATE message from the data packet format to JSON format and simplified the finite state machine. Each JSON message can carry multiple network view updates. In the normal condition, each NOS keeps keep the TCP connection by sending out KEEPALIVE messages to its peers in parallel. Once there are network view updates, one peer can directly push the UPDATE message to others at a high speed without re-setup TCP connections.

## 3. USE CASE – PATH COMPUTING APP

With EWBridge, each NOS can learn the global network view. For hosts in different SDN networks to communicate with each other, we design a path computing application shown in Figure 3. Each SDN network runs one path computing application. The application will compute and install flow paths in three SDN networks for all the outgoing packets whose destination IP addresses belong to peers' network.




| PP | VP | SLA |
|---|---|---|
| (S1, S2, S6) | (S1, S6) | latency = t (x1) |
| (S1, S3, S6) | (S1, S6) | cost = money (y1) |
| (S1, S4, S5, S6) | (S1, S6) | bd = bps (1G) |

| PP | VP | SLA |
|---|---|---|
| (S7, S8, S11) | S11 | bd(x2) |
| (S7, S8, S9, S10) | S11 | bd(y2) |

Fig. 3. physical view to virtual view abstraction (PP: Physical Path; VP: Virtual Path; OF: OpenFlow; S: Switch; bd: bandwidth; t: time; bps: bits per second

Two level designs of the path computing application:

(1) In the first step (around September 15), EWBridge exchanges the physical network view of the entire network information.

(2) This is a suggestion for the future implementation after the first step.

Some domains may be willing to expose only a part of the network view, rather than the whole network view, due to their privacy concerns. EWBridge permits abstracting a physical network to a virtual network for such domains.

As showed in Figure 3, routing paths from the ingress port to the egress port of an abstract network can have SLA-level path attributes such as time latency, reliability, bandwidth, packet loss rate like VP 1, VP 2, VP 3. To abstract a network further, EWBridge supports a network abstract to a virtual node like network 2. The virtual node for network 2 only retains the three physical inter-network links like link2, link3, link4.

Each path computing APP on NOS can compute the routing path (just a routing segment such as VP1 or VP2) within its domain. To compute a complete or global routing path, the NOS should at least know other abstracted virtual network view. The abstracted virtual network view is the minimum information for global network reachability. Then the path computing APP can compute and install a routing path by the restful API provided by the NOSes with the OpenFlow[8] protocol.

For the virtual network view, each NOS should store a mapping table between the physical network and the abstracted virtual network views which is showed as physical path to virtual path in Figure 3. To achieve large-scale scalability, each NOS can abstract a node such as network 2 with multi-ports and multiple bandwidth. Other NOSes treat the network domain 2 as an abstract node with 3 ports and 3 links.

## 4. IMPLEMENTION

We will implement the EWBridge and path computing application in one or two open source NOSes such as Floodlight. The position of EWBridge and path computing application is shown in Figure 4. The path computing application will be carried out in two steps as explained in Section 3.

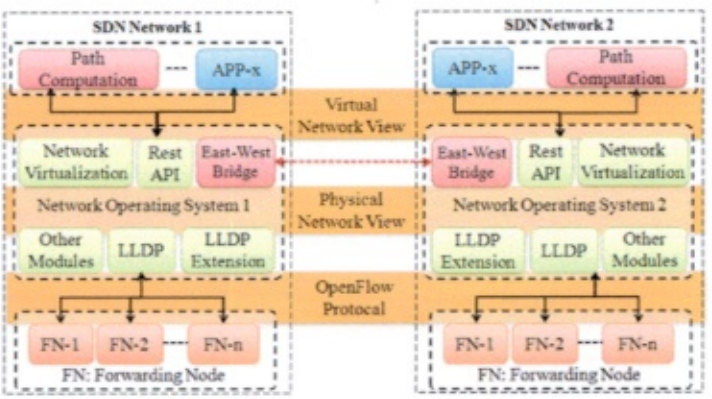



Fig. 4. EWBridge and path computatin application in different NOSes

## 5. REFERENCES


[1] N. McKeown. "Keynote talk: Software-defined networking". In Proc. of IEEE INFOCOM'09, Apr.2009.

[2] YAML: http://yaml.org/spec/1.2/spec.html.

[3] Floodlight:http://www.projectfloodlight.org/floodlight/.

[4] N. Gude, T. Koponen, J. Pettit, B. Pfa, M. Casado, N. McKeown, and S. Shenker. "NOX: Towards and operating system for networks". In ACM SIGCOMM Computer Communication Review, July 2008.

[5] YANG: https://tools.ietf.org/html/rfc6020.

[6] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski,M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. "Onix: a distributed control platform for large-scale production networks". In OSDI, 2010.

[7] A. Tootoocian and Y. Ganjali. "HyperFlow: A distributed control plane for OpenFlow". In INM/WREN workshop, 2010.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: enabling innovation in campus networks". ACM SIGCOMM Computer Communication Review, 38(2), 2008.