A collaborative model for routing in multi-domains OpenFlow networks

Xuan Thien Phan, Nam Thoai
Faculty of Computer Sciences and Engineering
Ho Chi Minh City University of Technology
Ho Chi Minh city, Vietnam
XuanThien.Phan@edu.hefr.ch
nam@cse.hcmut.edu.vn

Pierre Kuonen
Institute of Information and Communication Technologies
University of Applied Sciences of Western Switzerland of Fribourg
Fribourg, Switzerland
Pierre.Kuonen@hefr.ch

***Abstract*—OpenFlow is an innovative network architecture which decouple control plane and data plane, allowing researchers the ability to program their networks, control the behaviour of network switches for their experiments. However, current OpenFlow architecture relies completely on the use of a centralized controller to manage all switches connecting to it in the network. For networks with large number of switches, depending on a controller to manage all switches in the network become unfeasible. We believe that allowing multiple controllers distributedly manage OpenFlow network is the key factor for OpenFlow scalability. Current OpenFlow support allows to deploy muti-controllers networks in which each controller is responsible for operating each smaller domain in the networks. However, there is currently no mechanism for these controllers to associate with the each other. As a result, the controllers mostly have no idea how to process cross-domain packets.**

This paper proposes a collaborative model for the above multi-domains OpenFlow networks, provide the way for the controllers from different network domains to collaborate with each other in order to help them manage their networks more efficiently. Based on the model, we propose a solution for routing in multi-domains OpenFlow networks.

Keywords—Software Defined Networking; OpenFlow ; Routing; Scalability; Design ;

I. INTRODUCTION

Network infrastructure today mostly relies on devices (switch, router,…) whose platforms are closed by equipment vendors. The difficulty in making changes in these closed platforms is the huge obstacle for new ideas and innovations for the network development. Toward the goal to obtain an open and standard platform, Software Defined Network (SDN) [4,5] has been proposed enabling researchers the way to run their experiments in production networks. In SDN, the data plane and the control plane of the network are decoupled allowing network applications and services to be directly programmed in a remote controller to control the behaviours of network devices through a well-defined communication interface called OpenFlow [1,2,6,7,8].

Several controller platforms have been developed allowing researchers to write controller applications in different programming languages, such as NOX [12], Beacon [13], Onix [11], Floodlight [14], and Trema [15]. A variety of researches have been proposed using OpenFlow platform to solve different networking problems, like web server load-balancing [17], a solusion for DDoS flooding attact detection [19], and a network programming language [20]. OpenFlow is currently supported by a number of network equipment vendors with both hardwares (switches, routers, access points) and softwares (virtual switches) as well as deployment tools like Mininet [16].

However, current OpenFlow deployment currently doen't support multiple controllers in an OpenFlow network and this can be a lack of scalability. In addition, if switches in the network are geographically distributed over a wide area, the response time for requests of far switches becomes slow due to the latency of control bandwidth.

In this paper, we propose RMOF, a model that support multiple controllers concurrently manage the OpenFlow network. In RMOF, controllers are placed distributedly and each of them is responsible for operating a smaller domain of the network. The key factor of RMOF is a mechanism that allows controllers associate together, share their management information.

Why controllers need to exchange management information? With current OpenFlow support, each controller just manages assigned switches in the local network. It has almost no idea about the other switches existing in the other networks. As a result, when a controller need to send packets to a switch which belongs to another network, it doesn't know how to process. The controller mostly apply flooding mechanism to route those packets, and we believe this is not an optimal routing solusion.

In RMOF model, together with the information exchanging channel for controllers, data and applications were also designed as a solusion for routing over multiple network domains. Our routing method is a highly efficient one since it can compute optimal routes for packets travelling over multiple domains.

The remainder of the paper is organized as follows. In section II, we present in more detail about OpenFlow and provide an overview on related works. Section III explains

978-1-4673-2088-7/13/$31.00 ©2013 IEEE

about the proposed RMOF model, its elements and applications as well as how they work to solve the problem of routing across multiple network domains. In section IV, we give a conclusion of the paper and our plan for future work.

II. BACKGROUND AND RELATED WORKS

In this section, we present a brief introduction about main features of OpenFlow and the related works concerning our research. A more detailed description of OpenFlow and OpenFlow Switch specification is available at [1,7,9]. An OpenFlow network includes forwarding plane and control plane. The forwarding plane includes OpenFlow switches who are responsible for directly forward packets in the network in per-flow basis, and the control plane is an remote controller who is responsible for managing OpenFlow switches in the network.

Each OpenFlow switch contains one or more Flow Tables, which perform packet lookups and forwarding, and a Secure Channel to the controller (fig. 1). This is the communication channel which the controller use to manage all switches in the network via an OpenFlow protocol. Each table contains Flow Entries which are used as rules to apply to process matching packets. The controller can add, update or delete flow entries in flow tables and this is the way it control flow traffic in the network. Each flow entry mainly consists of match fields that defines the flow, instructions show what to do with matching packets, and counters for statistical purpose.

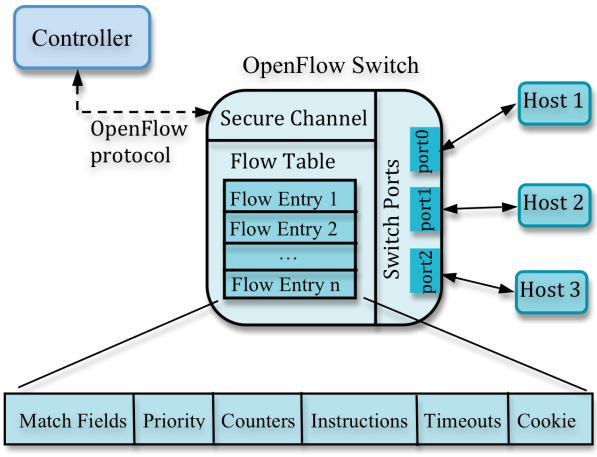

Figure 1. Architecture of an OpenFlow network.

When the OpenFlow switch receives a packet, it will lookup in its flow tables to find matching flow entry. If matching entry found, the switch applies instructions defined in the entry to process the packet and update counters. If no matching entry found, it sends the packet to the controller. The controller may add a new flow entry to the switch's flow table with instructions telling it how to process the packet, or simply drop the packet. This mechanism allow the controller to manage OpenFlow switches and control traffic in the OpenFlow network.

Some researches concering the concept of multiple controllers in OpenFlow network have been proposed recently. FlowVisor [3] allow multiple controllers in an OpenFlow network but its mechanism is to slice the network into separate slices and let each slice managed by a controller. These controllers are completely independent, each one doesn't affect the others and the goal of FlowVisor is to allow multiple researchers use the same network resources.

HyperFlow [10] has another approach. It raises the idea that enables interconnecting independently managed OpenFlow networks. HyperFlow uses multiple controllers to manage the networks and provide a publish/subcribe system where the controllers use to update the network state and send commands to the other controllers. However, this publish/subribe system bases on the use of a distributed file system, and the events or commands which are exchanged between the controllers and this system is done by a mechanism of reading or writing files in the file system. This seems not to be a reliable mechanism for exchanging networking information and the proposed design is just a general idea and doesn't solve any specific networking problem.

In the newest version 1.3 of OpenFlow switch specification [9], the idea about multiple controllers concept is presented. Depending on this idea, each OpenFlow switch can establish the connection with a single controller or with multiple controllers. With multiple controllers, the reliability is improved because the switches can continue to process packets in OpenFlow mode if they lose the connection with one controller. This idea allows multiple controllers manage an OpenFlow network but just the primary one is responsible to directly manage the all switches of the network, the other ones are almost used for recovery purpose to improve the reliability of the network. All of the controllers are aimed to take care of the same network and there is still no mechanism for the controllers to share their management works or collaborate together to manage the network.

III. RMOF MODEL

In our design, OpenFlow switches are used as forwarding plane for the network. These switches are managed by controllers placing distributedly in the network. Each switch connects to a nearest controller and is directly operated by that controller. Each controller together with switches connecting to it form a smaller OpenFlow network. We note that each network like that is considered as a network domain in the whole network including all controllers and switches in RMOF network (fig. 2). The controllers in RMOF are deployed by similar control platforms and networking applications to make sure that all switches will be operated by similar functionalities even when they are in different domains.

In RMOF model, a RMOF collaborator running on a separate computer was used as a communication channel for controllers from different network domains to associate with each other. The RMOF collaborator maintains a global view of the whole network. This global view is the information which is contributed by controllers of all domains in the network. In RMOF collaborator, together with the global view, applications are built to process the maintained information and provide helpful instructions or networking services for controllers. The information in this global view and the applications in RMOF collaborator are built according to the networking functionalities that user wish to implement for the whole network. By letting controllers independent in operating their local network, our model prevents the dependence of controllers on the RMOF collaborator. Hence it nearly minimizes the frequence of the

contact between controllers and the RMOF collaborator and just requires lightweight processing in the RMOF collaborator in a global level.

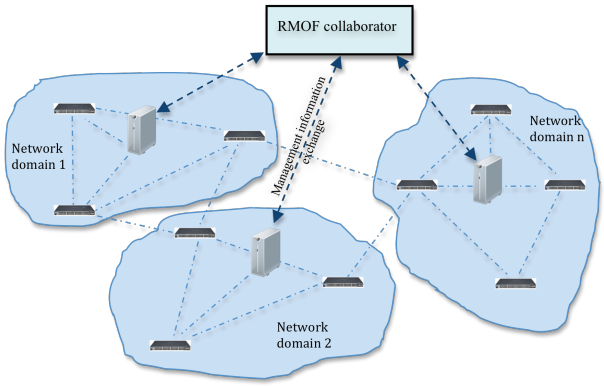

Figure 2. General design of RMOF model.

Our design allows new network functionalites and services to be easily implemented and deployed to deal with a variety of networking problems which solving them needs the collaboration of multiple networks, hence it is a promised solusion for OpenFlow scalability. In the following section, we describe in more detail about the global view and applications deployed in RMOF as well as how they work to improve routing packets over multiple network domains, the key factor to ensure the efficient operation for switches in the network.

A. Main components of RMOF model

Before presenting about components of RMOF, we describe in more detail about flooding mechanism that OpenFlow controllers currently apply for cross-domain packets as mentioned in the introduction section. As we know, when receiving a Packet-In from an OpenFlow switch, the controller replies by a Packet-Out telling the switch how to process that packet. For multi-domains networks managed by multiple controllers and with the current architecture, that happens just in case the packet's destination address is inside the domain managed by that controller. Other wise the controller doesn't know how to process that packet. In this case, the only one choice for the controller is to tell the switch to flood the packet to all of its ports except the incomming one. This process will be repeated for the neighbors of that switch and so on, with the hope that the packet will reach its destination. Realizing that flooding is not the good way to process packets travelling across mutiple domains, and basing on the above general model, we design specific data and elements with a proposed method to provide routing instructions for controllers to help them make better routing decisions.

In RMOF model, the communication channel (called RMOF channel) between controllers and the RMOF collaborator is done via a simple protocol designed for specific information exchange between them (fig. 3). In the RMOF collaborator, a Global Topology View was maintained based on the topology information of network domains shared by controllers and an application (called RMOF Global Routing App) to compute routes for packets travelling across mutiple domains (called cross-domain packets) based on the use of the Global Topology View. In each controller, an routing application (called RMOF Local Routing App) was added as a supplement for its existing routing application to process cross-domain packets, and a Route-Ins Table was deployed. This table is the place where the controller can lookup to find correspondent routing instructions to route cross-domain packets.

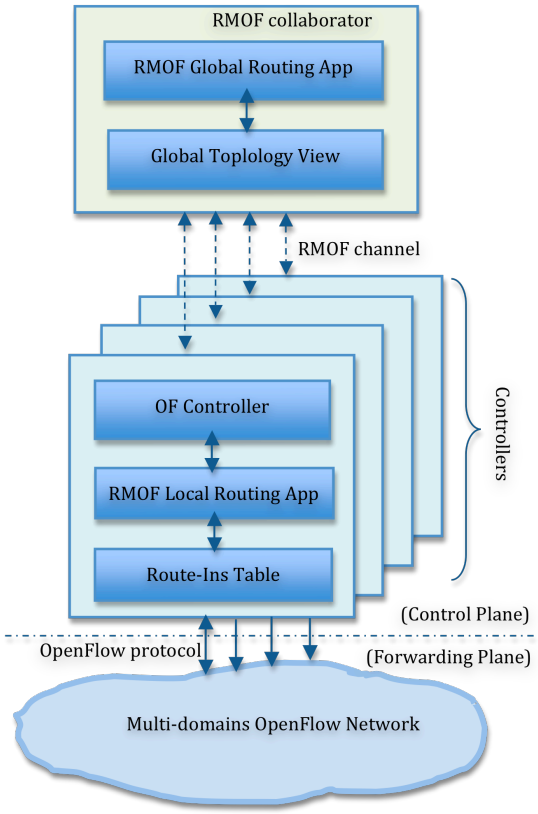

Figure 3. Main components of RMOF model.

1) The Global Topology View

The Global Topology View holds the information of all cross-domain links in the network and the estimated routing-costs between them contained in 2 tables:

a) Cross-domain Link Table

A Cross-Domain Link is the network link between 2 OpenFlow switches which belong to 2 diferrent network domains. This table contains the information of all cross-domain links of the network. Each entry of this table contains the information representing for a cross-domain link (fig. 4), including these fields:

- Src Controller ID: The ID of the controller who manage the source switch of the cross-domain link.
- Src Switch ID, Src Port: The ID and port of the source switch of the cross-domain link.
- Switch ID, Dst Port, Dst Controller ID: The switch ID, port, and controller ID of the destination switch of the cross-domain link. Notice that the role of the source switch and the destination switch of a cross-domain link are equal, so each cross-domain link will be presented by only 1 entry in the table.

Src Controller ID	Src Switch ID	Src Port	Dst Port	Dst Switch ID	Dst Controller ID

Figure 4. Fields of a Cross-domain Link Entry

b) Estimated Routing-cost Table

An estimated routing-cost is the routing-cost of the shortest path between a pair of cross-domain OpenFlow switches in a network domain. This routing-cost is counted by the number of direct network links (or by the number of switches – 1) of the shortest path between those cross-domain switches computed the controller who manage that domain. The Estimated Routing-Cost Table is used to save the data about the routing-costs of all pairs of cross-domain switches of all domains in the network. Each entry in this table presents for an estimated routing-cost between each pair of cross-domain switches (fig. 5), including these fields:

- Controller ID: The ID of the controller who manages the network domain.
- Src Switch ID, Dst Switch ID: The IDs of the a pair of cross-domain switches in the domain managed by the above controller.
- Estimated routing cost: The routing cost between the above pair of cross-domain switches.

Controller ID	Src Switch ID	Dst Switch ID	Estimated routing-cost

Figure 5. Fields of an Estimated Routing-cost Entry

Each controller in the network is responsible to send switch IDs of all cross-domain switches in its domain and computed routing-costs between these switches to the RMOF collaborator, where these information will be processed together with the information from the other network domains to form the Global Topology View. This view is alike a 'weighted graph' in which nodes are cross-domain switches, edges are computed paths between those switches, and weights are estimated routing-costs between them (fig. 6).

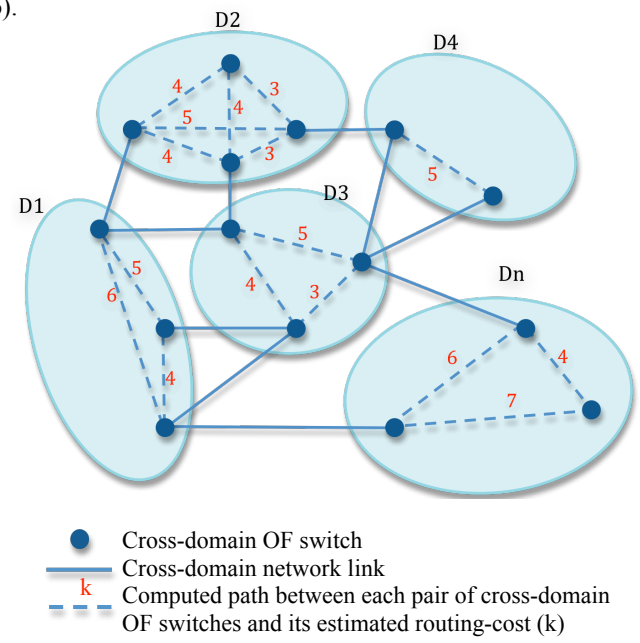

Figure 6. An image of the Global Topology View seen from RMOF collaborator

2) Route-Ins Table

In RMOF, each controller is proposed to hold a Route-Ins Table. It is a learning table which contains routing instructions that help the controller in routing cross-domain packets. Each entry in this table supplies enough information showing the controller how to route cross-domain packets (fig. 7), including these fields:

- Local Src Switch ID: The ID of the source switch in the local domain.
- Terminal Dst Address: The packet's destination address.
- Terminal Dst Switch ID: The ID of the destination switch.
- Terminal Dst Controller ID: The ID of the destination domain's controller.
- Local Dst Switch ID: The ID of the cross-domain switch in the current domain that the controller needs to route the packet to.
- Local Dst Out-Port: The outgoing port number of the above Local Dst Switch, it belongs to a cross-domain link where packets will be forwarded through to the next domain.
- Estimated Routing-cost: The total estimated routing-cost of the route from the local source switch of current domain to packet's destination address.

Local Src Switch ID	Terminal Dst Addr	Terminal Dst Switch ID	Terminal Dst Controller ID	Local Dst Switch ID	Local Dst Out-Port	Estimated Routing-cost

Figure 7. Fields of a Route-Ins Entry

Each entry in this table instructs the controller to route destination-matched packets to the local switch whose ID is Local Dst Switch ID, and then forward packets out of the port Local Dst Outgoing Port of this switch to the next network domain. Similar processes happen in next network domains, and by this mechanism packets are forwarded through domains to their desired destination.

The Route-Ins Table is the key factor to improve routing over multiple network domains, since it helps controllers route cross-domain packets via shortest routes computed by the RMOF collaborator instead of flooding them. In case the controller doesn't find any entry in the table that matchs the packet's destination, it can contact the RMOF collaborator to ask for routing instruction. The routing instruction it receives from the RMOF collaborator then will be added to the Route-Ins Table for subsequent use.

3) RMOF Local Routing and RMOF Global Routing applications

The RMOF Local Routing application placing at the controller side is responsible for discovering cross-domain links in its domain and computing routing-costs between them in order to report to the RMOF collaborator. It also takes on adding or deleting entries in Route-Ins Table, looking up in Route-Ins Table for matched entries, and computes local routes for cross-domain packets based on matched routing instructions, as well as contacts the RMOF collaborator for routing instructions when it doesn't find any matched Route-Ins entry.

The RMOF Global Routing application placing at the RMOF collaborator is responsible for building up the Global Topology View based on discreted topology information it receive from controllers, and computing global routes for

cross-domain packets. These routing applications collaborate with each other to compute routes for cross-domain packets and the algorithm used to compute routes is the Dijkstra's shortest path algorithm [18].

B. *Discorvering cross-domain network links*

In this section, we explain how cross-domain network links are discovered based on the topology information shared by controllers.

In a network domain managed by a controller, after the connection between the controller and the OpenFlow (OF) switches was established sucessfully, each of these switches sends out a LLDP packet to all of its neighbor switches. This message contains the sending switch's ID together with its outgoing port number. When receiving a LLDP packet from one its neighbor switches, the recipient encapsulates and sends it to the controller. In the controller, the sending switch ID field in this LLDP message will be extracted and checked if it matches any known switch.

- If it matches a known OF switch in the controller's domain, this means there is a network link between 2 known switches (with source switch ID and destination switch ID specified in the LLDP message) in its network domain, the controller will learn that link for future processing. This is the current processing of current support of OpenFlow controller to discover all network links between all pairs of OF switches in its domain.

- If the source Switch ID of a LLDP message doesn't match any known OF Switch in that domain, the current OpenFlow controller doesn't have any further processing in this situation.

In the proposed RMOF model, for the above second case, the controller sends this information to the RMOF collaborator. The information contains fields of the LLDP message it received from the local switch: a destination switch ID, destination incomming port number (of the known switch in its domain) and the source switch ID, source outgoing port (of a switch somewhere in the world outside the controller's domain that it has no idea about). When receiving that information from the controller, together with the other information receiving from the other controllers, the RMOF collaborator will combine them to build up the Cross-domain Link Table.

C. *Computing routing-costs between cross-domain switches in a network domain*

This section describes how controllers compute routing-costs between cross-domain switches in their local network domains. When an OpenFlow switch in a network domain receives a LLDP message from an unknown OF Switch outside its domain which the local controller has no idea about (this internal OpenFlow Switch in this current domain is called 'cross-domain switch'), before sending this network link information to the RMOF collaborator, the controller computes the routing-costs of the paths from this cross-domain switch to the other known cross-domain switches in its domain. These information will be encapsulated and sent to the RMOF collaborator to help it buid the Estimated Routing-cost Table. A controller who manage a network domain has enough information to do so because it knows all of switches in its domain as well as all the direct network links between them. The 'local view' that the controller observes in its domain is alike a weighted graph as showed in fig. 6. With these information, the controller can apply an algorithm to compute the path between 2 switches in its domain. In routing applications of the model, the Dijkstra's Shortest Path Algorithm [18] was used to compute the shortest path between those switches.

D. *Computing shortest path routes and routing cross-domain packets*

In this section, we explain how the routing applications compute routes for cross-domain packets and route them. When the controller in a network domain receives a Packet-In from one of OpenFlow switches it manages, first it extracts the packet's header. If the packet's destination is inside the controller's network domain, the controller processes it normally as specified in OpenFlow Switch specification v1.3 [9] (replies by Packet-Out, drops the packet, …).

If the controller has no idea about the packet's destination (in case of cross-domain packets where their destinations belong to other network domains outside the current controller's domain), it passes the packet to the RMOF Local Routing application. This application first looks up in Route-Ins Table for destination-matched entry. If an entry found, the application will route the packet based on the information in this entry. In case no destination-matched entry found, the application contacts the RMOF collaborator to ask for routing-instruction.

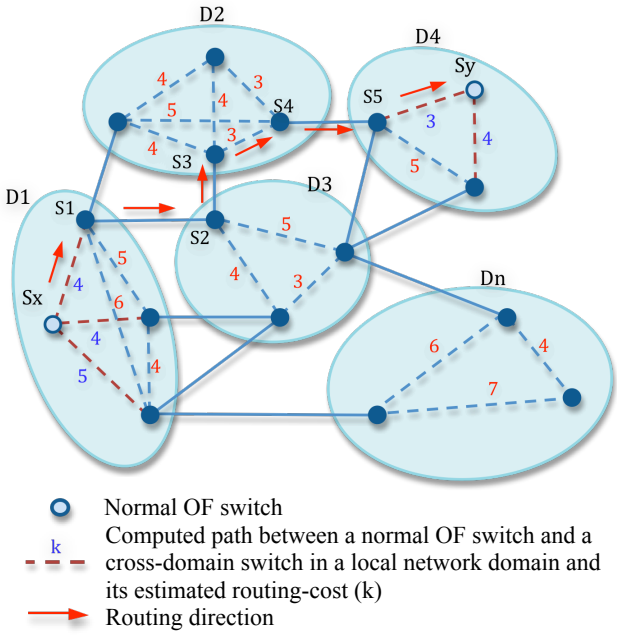

Figure 8. An example of global route computing in RMOF collaborator.

First, the controller computes routing-costs between the source OF Switch (switch Sx in fig. 8) to all of the cross-domain switches in its domain, and then send them together with the information of packet's destination to the RMOF collaborator. In RMOF collaborator, the RMOF Global Routing application locates the packet's destination by asking the other controllers except the one whom it received routing-instruction request from. Replies from controllers help it know where packet's destination is (switch Sy in fig. 8) and also the estimated routing-costs from that destination

to cross-domain switches in destination domain (computed by the controller who manages it).

Now the RMOF Global Routing application has enough information to compute a global route for the packet. What it observes is alike a 'weighted graph' in which nodes are switches (including all cross-domain switches, the normal source switch and destination switch), edges are computed paths between them, and weights are estimated routing-costs of these paths (fig. 8). The application computes a global route based on those information, and this is the optimal one due to the use of Dijkstra's algorithm (for example the route Sx - S1 - S2 - S3 - S4 - S5 - Sy in fig. 8).

Then routing-instructions are sent to correspondent controllers along the computed route, and those instructions will help them route the packet and subsequent destination-matched packets in shortest way. For example in fig. 8, packets will be routed from switch Sx to switch S1 in the source network domain D1, then forwarded through the cross-domain link S1<->S2 to switch S2 of network domain D3. Similar routing processes happen in the network domain D3, D2 until packets reach switch S5 of destination network domain D4. Here the controller in D4 know how to route packets from S5 to Sy and forward out the destination host. We note that controllers route packets by directly adding Flow entries to Flow Tables of OpenFlow switches along the routes computed by RMOF routing applications.

IV. Conclusion and Future Work

In this paper, we proposed RMOF model which allows multiple controllers in different OpenFlow networks associate with each other, and a solusion for routing cross-network packets based on the model. By letting controllers independently manage their local networks, RMOF keep all advantages of OpenFlow architecture while providing a mechanism allowing them to exchange necessary information to make the networks operate more efficiently. Our design is a promised solusion for OpenFlow scalability since it enables control planes of mutiple OpenFlow networks collaborate together to operate networks as a whole one. It's also reliable because the communcation between controllers and the RMOF collaboratior bases on the use of a secure channel. Our model can be used to improve networking functionalities due to the global view maintained as an observation over all networks. The routing solusion we presented is a case study for the applicability of our model. The global view holds topology information about cross-domain network links together with estimated routing-costs between them, hence routing applications have enough information to calculate routes for cross-domain packets. Our solusion appreciably improves the functionality of routing over multiple networks because computed routes are optimal ones due to the use of a shortest path algorithm.

We are in working progress to provide a complete implementation for our design as well as run experiments in a large testbed to evaluate its efficiency, performance and scalability. In the future we also plan to extend the model to deal with the other networking problems which need the collaboration of multiple controllers to solve such as load-balancing over multiple networks. The direction is to extended the Global Topology View, allowing controllers to share information concerning their netwoks more completely, including other information such as the traffic load of partner networks, instead of just topology information as currently.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008.

[2] "Intro To OpenFlow," 2011. [Online]. Available: https://www.opennetworking.org/standards/intro-to-openflow

[3] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, Guru Parulkar, "Flow Visor: A Network Virtualization Layer," Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks, 2009.

[4] "Software-Defined Networking: The new norm for networks," Open Networking Foundation, April 2012.

[5] Mark Reitblatt, Nate Foster, Jennifer Rexford, David Walker, "Software-Defined Networks: Change you can believe in," In *Hotnets*, 2011.

[6] "OpenFlow Switch Specification v1.1.0," 2008. Available: http://www.openflow.org/wp/documents/

[7] Open Networking Foundation. https://www.opennetworking.org/

[8] A"OpenFlow Switch Specification v1.2," Open Networking Foundation, December 2011. Available: https://www.opennetworking.org/images/stories/downloads/specification/openflowspecv1.2pdf

[9] "OpenFlow Switch Specification v1.3.0," Open Networking Foundation, 2012. Available: https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf

[10] Amin Tootoonchian, Yashar Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," In Proceedings of NSDI Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN), San Jose, CA, April 2010.

[11] T. Koponen, M. Casado, N. Gude, J. Stribling, P. L., M. Zhu, R. Ramanathan, Y. Iwata, H. Inouye, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," In *Operating Systems Design and Implementation*. USENIX, 2010.

[12] N.Gude, T. Koponen, J. Pettit, B. Plaff, M. Casado, and N. McKeown, "Nox: Towards an operating system for networks," In *ACM SIGCOMM CCR: Editorial note*, July 2008.

[13] "Beacon: Java-based OpenFlow Control Platform," 2012. [Online]. Available: http://www.beaconcontroller.net/

[14] "Floodlight OpenFlow Control Platform," 2012. [Online]. Available: http://floodlight.openflowhub.org/

[15] "Trema OpenFlow Control Platform," 2012. [Online]. Available: http://trema.github.com/trema/

[16] a Bob Lantz, Brandon Heller, Nick McKeown. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Hotnets*, pages 1-6, 2010.

[17] Richard Wang, Dana Butnariu, Jennifer Rexford. OpenFlow-based Server load-balancing gone wild. In *Hot-ICE*, Mar 2011.

[18] "Dijkstra's shortest path algorithm," 2012. [Online]. Available: http://en.wikipedia.org/wiki/Dijkstra's_algorithm

[19] Rodrigo Braga, Edjard Mota, Alexandre Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," In *35th Annual IEEE Conference on Local Computer Networks*, Colorado, USA, 2010.

[20] N. Foster, R. Harrision, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: a network programming language," In *ICFP*, Japan, September 2011.