A SDN Network Based on VLANs for Data Centers

Fei XU
Shanghai Engineering Research Center for Broadband Networks and Applications
Shanghai, China

Yanwei XU
Shanghai Engineering Research Center for Broadband Networks and Applications
Shanghai, China

***Abstract*—The demands for the scalable and efficient network at data centers are becoming more and more intense recently. However, there are three problems needed to be solved, i.e., the ever-increasing number of actual or virtual machines demands, effectively managing these different machines and maintaining the effectiveness. This paper proposes a new software defined network (SDN) based on VLANs and its significant characteristic is to build a scalable layer 2/3 network for data centers based on the physical OpenFlow switches. Our framework has high performances because: 1) The operation of encapsulating and unpacking packets is not necessary; 2) Routing paths can be pre-computed.**

Keywords—Data Centers; SDN; OpenFlow; Scalable Layer 2/3 Network;

I. Introduction

There is an increasing trend toward migrating applications, computation and storage of data centers, which is fashionable across the Internet. Benefiting from commodities of scale leads to the emergence of "mega data centers" that is able to make host applications running on tens of thousands of servers [1]. For instance, a web search request may access an inverted index spread across 1,000+ servers, and correspondingly, data storage and analysis applications may interactively process petabytes of information stored on thousands of machines. There are numerous networking application requirements across all these cases. In order to overcome the disadvantages of the heavy administrative overhead and support end host virtualization, most of the data centers deploy a layer 2 network where forwarding is performed based on flat MAC addresses. However, the traditional layer 2 protocols such as Spanning Tree Protocol (STP) are not suitable for a large data center network at this scale because of the long data interruption in STP recalculations, inefficient utilization of the available bandwidth, broadcast storms and so on. Therefore, the design of a novel data center network architecture with the features of scalability, low cost, robustness, and energy conservation is required.

Software defined network (SDN) [6, 7, 8] may be another effective answer to solve the problem of large-scale network across data centers. SDN is an emerging paradigm in computer networking that allows a logically centralized software program to control the behavior of an entire network. But a navie SDN technology is still difficult to solve those problems mentioned above, the reasons are: (1) the configurations of controller in a pure SDN is very high, thus it is difficult to provide excellent performance for large-scale networks; (2) its scalability will become very poor due to the restrictions of the supported maximum number of flow entries of the commercial physical switches.

In this article, we share the experience of developing a SDN-based data center network, which constructs a switch overlay to enable packet routing among switches based on the VLANs that indicates the destination OVS switch of the packet. The shortest paths among the OVS switches should be computed when constructing the overlay at the beginning, so they do not need to be computed any more while communicating between any pair of hosts. The MAC address resolving and the rules to push VLAN IDs for the packets can be done by a cluster of Controllers to achieve high efficiency and scalability. Not only can the proposed network achieve the intrinsic flexibility of SDN, but also it has the high efficiency and scalability due to its distributed switching, small number of flow entries in each switch, pre-computed paths, and low interactions between the control and data planes.

This paper is organized as follows: related works are described in section 2. In section 3, the proposed framework and problem definition are presented. Finally, concluding remarks are given in section 4.

II. Related work

Software-defined network (SDN) has become one of the most important architectures for the management of large scale, complex networks which may require re-policing or reconfigurations from time to time. SDN achieves easy re-policing by decoupling the control plane from data plane. Thus the network routers/switches just simply forward packets by following the flow table rules set by the control plane. Currently, OpenFlow is the most popular SDN protocol/standard and has a set of design specifications. Although SDN/OpenFlow is a relatively new area, it has attracted much attentions from both academia and industry. In SDN, the Controller works as an ARP-proxy to handle the ARP requests since it has the full knowledge of the whole network topology and the exact location of every host. And the switches route packets under the instructions of the installed flow entries. Therefore, the two main challenges in designing large layer 2 networks can be solved inherently.

A typical communication between any pair of hosts, named as h_1 and h_2, in current popular SDN platforms such as OpenDayLight (ODL) [2] and Ryu [3], is shown as follows:

1) If h_1 wants to send a packet destined to h_2, the packet will be transmitted to the ingress switch of h_1 at first. And then the switch forwards it to the Controller.

2) The Controller compute a suitable path for h_1 and h_2 based on the network topology, and installs corresponding OpenFlow flow entries to the switches along the computed path from h_1 to h_2. Then, the subsequent packets are routed according to the installed flow entries.

Nevertheless, such procedures have too many interactions between the Controller and switches. And a large number of flow entries are required in the switches. Although SDN can work in the proactive operation mode and its switch flow tables can be populated in advance for predicting all the possible traffic in a cloud data center, it still can involve huge communications and lead to very large number of flow entries, which is unrealistic for hardware. In spite of providing coarse flow granularity, flow aggregation can be a possible remediation for scalability bottlenecks in the core network. However, the wildcard entries space that supports flow aggregation is usually rather limited, even in flexible software implementations (e.g., 100 entries reference the Stanford implementation). Recently, [13] and [14] have proposed that core networks can benefit from independent evolution between switching fabric and edge nodes in SDN. In other words, the way of a switch fabric to provide raw forwarding capacity to interconnect the edge nodes can be beneficial for the speed and cost. However, they are purely conceptual proposals.

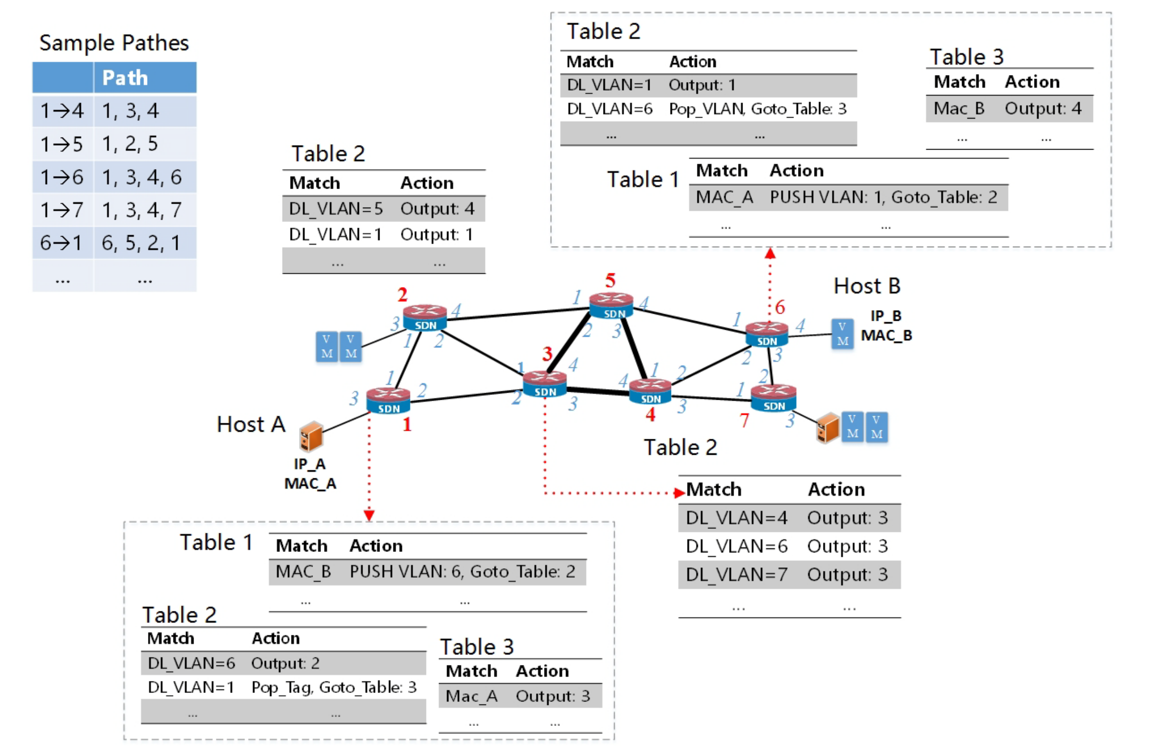
Compared to the above solutions, the proposed network in this paper achieves the intrinsic flexibility of SDN, and has the high efficiency and scalability due to its distributed switching, highly reduced number of flow entries in switches, pre-computed paths, and low interactions between the Controller and switches.

III. Problem Definition and the Framework

The architecture of the proposed network is same to the standard architecture of the OpenFlow [9] based SDN. The core is the SDN Controller, which controls the actions of all the switches via OpenFlow messages. Each switch transmits data packets according to the installed flow entries in its flow tables. The main characteristics that differentiate it from the existing SDN solutions are the two stages in constructing paths for hosts and routing packets in switch level.

After discovering the network topology, the Controller computes paths for all pairs of switches and assigns a tag for each path. Then, each path is installed into the related switches along the path via OpenFlow flow entries. Note that the paths does not have to be the shorted paths and can be computed arbitrarily. The above procedures are called as the *Fabric construction*, which is basis of the following unicasting path construction. For instance, suppose that switches 1, 3, 4 and 6 are on the path from switch 1 to switch 6 in Fig.1. Then, in switch 1, the data packets destined to the hosts connected to switch 6 will be transferred to switch 3 via port 2. The main challenge in implementing Fabric Construction is the dynamic in the network topology: switches and links all can be added or removed. After the Controller notifies such events, it must compute the impacts and repair the affected paths.

Since OpenFlow can support multiple flow tables from



Sample Pathes

	Path
1→4	1, 3, 4
1→5	1, 2, 5
1→6	1, 3, 4, 6
1→7	1, 3, 4, 7
6→1	6, 5, 2, 1
…	…

Table 2

Match	Action
DL_VLAN=5	Output: 4
DL_VLAN=1	Output: 1
…	…

Table 2

Match	Action
DL_VLAN=1	Output: 1
DL_VLAN=6	Pop_VLAN, Goto_Table: 3
…	…

Table 3

Match	Action
Mac_B	Output: 4
…	…

Table 1

Match	Action
MAC_A	PUSH VLAN: 1, Goto_Table: 2
…	…

Table 2

Match	Action
DL_VLAN=4	Output: 3
DL_VLAN=6	Output: 3
DL_VLAN=7	Output: 3
…	…

Table 1

Match	Action
MAC_B	PUSH VLAN: 6, Goto_Table: 2
…	…

Table 2

Match	Action
DL_VLAN=6	Output: 2
DL_VLAN=1	Pop_Tag, Goto_Table: 3
…	…

Table 3

Match	Action
Mac_A	Output: 3
…	…

Fig.1: Sample Network Topology and the Flow Entries

version 1.1, we build up a packet processing pipeline of four flow tables in each switch for implementing the proposed method. Table 0 is responsible for packets preprocessing, which will redirect the packets to different steps of the pipeline. It has *p*+2 flow entries, where *p* is the quality of the switch ports that connects with other switches. The first *p* entries are as “in_port=*i*, ip, actions=goto_table: 2”, which are responsible for redirecting the packets from other switches to flow Table 2. And the left two flow entries are “ip, actions=goto_table: 1” and “arp, actions=Controller”, which are used for redirecting the packets from local host to Table 1 and the Controller, respectively. Table 1 stores the flow entries for tagging packets that are sent by local hosts to other hosts. Therefore, Table 1 is empty after the Fabric construction, whose flow entries are session-based and are installed in following unicasting path construction. Table 2 is responsible for forwarding packets according to the switch ids. It has at most *s* flow entries. The first *s*-1 entries output the packets with different tags to the corresponding ports, and are as “dl_vlan=i, actions=output: j”, where *s* is the total number of switches in the networks, and *i* is one of the VLAN id and *j* is one of port of local switch. The last flow entry, which removes the tags of the packets that are sent to the local hosts and then redirect them to Table 3, is as “dl_vlan=k, actions=pop_vlan, goto_table: 3$”, where *k* is the switch id of the local switch. Table 3 is responsible for the packets that are sent to the local hosts of the switch, which will output the packets to the right local ports according to the Mac or IP addresses. Table 3 is also empty after the initialization of SFabric construction; and a flow entry is installed once a local host is discovered.

Let's considering the switch with id 1 in Fig.1 as an example. Suppose the paths computed for switch 1 to 4, 5, 6 and 7 are shown as the table at the top left corner in Fig.1. Then the flow tables of switch 1, 3 and 6 would be as the tables shown in Fig.1. For instance, the fourth flow entry of switch 3 means that the packets with VLAN 6 should be outputted to port 3

When a host *A* wants communicate with a host *B* at the first time, it send an ARP request message to its access switch. If *B* is directly connected to A's access switch, the ARP request is resolved in the traditional manner; otherwise, the switch will forward the ARP request message to the Controller or the other switches, determined by the corresponding priorities of the flow entries.

If the Controller received the ARP request message, it will search the MAC address of *B* and the ID of its access switch in its database that storing the topology. And then, the Controller sends an ARP responding message to *A* and two OpenFlow messages to the two access switches of *A* and *B* (switch 1 and 6) for loading the two flow entries as shown in the talbes of Fig.1, whose purposes are to tag the data packets using the destination switch IDs. Note that such flow entries must be stored in the flow table prior to the table for storing the entries that are created in the booting up stage.

After the ARP resolving is finished, the data packets will be tagged by the destination switch by the access switch of the source host. And then, the data packets are forwarded in the switch overlay based on the destination switch ID.

IV. CONCLUSION

In this paper, we propose a large layer 2 network based on SDN technologies for data centers. A central controller is used to manage the routing maps among the switches. But different from other SDN solutions, it employs a totally different operation style for the Controller, which highly releases its workload, forwards flows more efficiently and solves the scalability problem.

In the future, we will test our framework in the real network and promote its performance. And the traffic engineering components will be developed and studied in the future. Furthermore, the ARP resolving is only done by the Controller, which will pose a heavy burden to the Controller. How to relieve such burdens would be another future study direction.

ACKNOWLEDGMENT

We would like to thank the Shanghai Technical Standards for Special Projects "Research on Future Network Architecture based on SDN" (14DZ0503000).

REFERENCES

[1] “Inside Microsoft's 550 Million Mega Data Centers," www.informationweek.com/news/hardware/data_centers/showArticle.html?articleID=208403723

[2] OpenDayLight.http://OpenDayLight.org.

[3] Ryu,' http://osrg.github.io/ryu/.

[4] OpenVSwitch. http://openvswitch.org.

[5] Programmable flow networking. http://www.necam.com/SDN/.

[6] O.N. Foundation. Software-Defined Networking: The New Norm for Networks. White paper. 2013.

[7] S.Jain, A.Kumar, S.Mandal, J.Ong, L.Poutievski, A.Singh, S.Venkata, J.Wanderer, J.Zhou, M.Zhu, J.Zolla, U.Holzle, S.Stuart, and A.Vahdat. B4: experience with a globally-deployed software defined wan. In ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013, pages 3--14, 2013.

[8] N.McKeown, T.Anderson, H.Balakrishnan, G.M. Parulkar, L.L. Peterson, J.Rexford, S.Shenker, and J.S. Turner. Openflow: enabling innovation in campus networks. Computer Communication Review, 38(2):69--74, 2008.

[9] B.~Heller, OpenFlow Switch Specification, Version 1.5.0

[10] A.Voellmy and J.Wang. Scalable software defined network controllers. In ACM SIGCOMM 2012 Conference, SIGCOMM'12, Helsinki, Finland - August 13 - 17, 2012, pages 289--290, 2012.

[11] A.Voellmy, J.Wang, Y.R. Yang, B.Ford, and P.Hudak. Maple: simplifying SDN programming using algorithmic policies. In ACM SIGCOMM 2013 Conference, pages 87--98, 2013.

[12] Y.Zhuang, L.Law, A.Rafetseder, L.Wang, I.Beschastnikh, and J.Cappos. Sensibility testbed: An internet-wide cloud platform for programmable exploration of mobile devices. In 2014 Proceedings IEEE INFOCOM Workshops, Toronto, ON, Canada, April 27 - May 2, 2014, pages 139--140, 2014.

[13] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, Fabric: A retrospective on evolving sdn," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2012, 2012, pp. 1—5.

[14] M. Martinello, M. R. N. Ribeiro, R. E. Z. de Oliveira, and R. de Angelis Vitoi, Keyflow: a prototype for evolving SDN toward core network fabrics,' IEEE Network, vol.~28, no.~2, pp. 12--19, 2014.